



## Automating Production of Cross Media Content for Multi-channel Distribution

[www.AXMEDIS.org](http://www.AXMEDIS.org)

### DE2.1.1.2.2 Use Cases and Scenarios, First Update of DE2.1.1b

**Version:** 3.0

**Date:** 25/03/2006

**Responsible:**

Project Number: IST-2-511299 Project Title: Automating Production of Cross Media Content for Multi-channel Distribution Deliverable Type: public Visible to User Groups: Yes Visible to Affiliated: Yes Visible to Public: Yes
---

Deliverable Number: DE2.1.1.2.2 Contractual Date of Delivery: see annex I Actual Date of Delivery: 25/3/2006 Work-Package contributing to the Deliverable: WP2 Task contributing to the Deliverable: all of WP2 Nature of the Deliverable: document Author(s): all mentioned in the section headings
--

**Abstract:**

This document reports the use cases collected for the realization of the AXMEDIS Framework and AXMEDIS tools in general for the automated production, protection and cross channel distribution of digital content.

**Keyword List:** Requirements, Multimedia, cross media, Cross channel distribution, content production, protection.

# AXMEDIS Copyright Notice

The following terms (including future possible amendments) set out the rights and obligations licensee will be requested to accept on entering into possession of any official AXMEDIS document either by downloading it from the web site or by any other means.

Any relevant AXMEDIS document includes this license. PLEASE READ THE FOLLOWING TERMS CAREFULLY AS THEY HAVE TO BE ACCEPTED PRIOR TO READING/USE OF THE DOCUMENT.

## 1. DEFINITIONS

- i. "**Acceptance Date**" is the date on which these terms and conditions for entering into possession of the document have been accepted.
- ii. "**Copyright**" stands for any content, document or portion of it that is covered by the copyright disclaimer in a Document.
- iii. "**Licensors**" is AXMEDIS Consortium as a de-facto consortium of the EC project and any of its derivations in terms of companies and/or associations, see [www.AXMEDIS.org](http://www.AXMEDIS.org)
- iv. "**Document**" means the information contained in any electronic file, which has been published by the Licensor's as AXMEDIS official document and listed in the web site mentioned above or available by any other means.
- v. "**Works**" means any works created by the licensee, which reproduce a Document or any of its part.

## 2. LICENCE

1. The Licensor grants a non-exclusive royalty free licence to reproduce and use the Documents subject to present terms and conditions (the **Licence**) for the parts that are own and proprietary property the of AXMEDIS consortium or its members.
2. In consideration of the Licensor granting the Licence, licensee agrees to adhere to the following terms and conditions.

## 3. TERM AND TERMINATION

1. Granted Licence shall commence on Acceptance Date.
2. Granted Licence will terminate automatically if licensee fails to comply with any of the terms and conditions of this Licence.
3. Termination of this Licence does not affect either party's accrued rights and obligations as at the date of termination.
4. Upon termination of this Licence for whatever reason, licensee shall cease to make any use of the accessed Copyright.
5. All provisions of this Licence, which are necessary for the interpretation or enforcement of a party's rights or obligations, shall survive termination of this Licence and shall continue in full force and effect.
6. Notwithstanding License termination, confidentiality clauses related to any content, document or part of it as stated in the document itself will remain in force for a period of 5 years after license issue date or the period stated in the document whichever is the longer.

## 4. USE

1. Licensee shall not breach or denigrate the integrity of the Copyright Notice and in particular shall not:
  - i. remove this Copyright Notice on a Document or any of its reproduction in any form in which those may be achieved;
  - ii. change or remove the title of a Document;
  - iii. use all or any part of a Document as part of a specification or standard not emanating from the Licensor without the prior written consent of the Licensor; or
  - iv. do or permit others to do any act or omission in relation to a Document which is contrary to the rights and obligations as stated in the present license and agreed with the Licensor

## 5. COPYRIGHT NOTICES

1. All Works shall bear a clear notice asserting the Licensor's Copyright. The notice shall use the wording employed by the Licensor in its own copyright notice unless the Licensor otherwise instructs licensees.

## 6. WARRANTY

1. The Licensor warrants the licensee that the present licence is issued on the basis of full Copyright ownership or re-licensing agreements granting the Licensor full licensing and enforcement power.
2. For the avoidance of doubt the licensee should be aware that although the Copyright in the documents is given under warranty this warranty does not extend to the content of any document which may contain references or specifications or technologies that are covered by patents (also of third parties) or that refer to other standards. AXMEDIS is not responsible and does not guarantee that the information contained in the document is fully proprietary of AXMEDIS consortium and/or partners.
3. Licensee hereby undertakes to the Licensor that he will, without prejudice to any other right of action which the Licensor may have, at all times keep the Licensor fully and effectively indemnified against all and any liability (which liability shall include, without limitation, all losses, costs, claims, expenses, demands, actions, damages, legal and other professional fees and expenses on a full indemnity basis) which the Licensor may suffer or incur as a result of, or by reason of, any breach or non-fulfilment of any of his obligations in respect of this Licence.

## 7. INFRINGEMENT

1. Licensee undertakes to notify promptly the Licensor of any threatened or actual infringement of the Copyright which comes to licensee notice and shall, at the Licensor's request and expense, do all such things as are reasonably necessary to defend and enforce the Licensor's rights in the Copyright.

## 8. GOVERNING LAW AND JURISDICTION

1. This Licence shall be subject to, and construed and interpreted in accordance with Italian law.
2. The parties irrevocably submit to the exclusive jurisdiction of the Italian Courts.

### **Please note that:**

- You can become affiliated with AXMEDIS. This will give you the access to a huge amount of knowledge, information and source code related to the AXMEDIS Framework. If you are interested please contact P. Nesi at [nesi@dsi.unifi.it](mailto:nesi@dsi.unifi.it). Once affiliated with AXMEDIS you will have the possibility of using the AXMEDIS specification and technology for your business.
- You can contribute to the improvement of AXMEDIS documents and specification by sending the contribution to P. Nesi at [nesi@dsi.unifi.it](mailto:nesi@dsi.unifi.it)
- You can attend AXMEDIS meetings that are open to public, for additional information see [WWW.AXMEDIS.org](http://WWW.AXMEDIS.org) or contact P. Nesi at [nesi@dsi.unifi.it](mailto:nesi@dsi.unifi.it)

# Table of Content

<b>1</b>	<b>EXECUTIVE SUMMARY AND REPORT SCOPE .....</b>	<b>12</b>
<b>2</b>	<b>STRUCTURE OF USE CASES.....</b>	<b>14</b>
2.1	STRUCTURE OF USE CASES .....	14
2.2	USE CASE AND SCENARIO DIAGRAM: SHAPES AND SEMANTICS .....	14
<b>3</b>	<b>GENERAL USE CASES .....</b>	<b>16</b>
3.1	MACRO-FUNCTIONALITIES.....	17
3.1.1	Automatic collection of content into local AXMEDIS Database from proprietary CMS .....	17
3.1.2	Querying for AXMEDIS objects and Selection creation.....	18
3.1.3	Automatic load (and update) of AXMEDIS objects into local AXDB from AXEPTool.....	21
3.1.4	Automatic protection of AXMEDIS objects .....	23
3.1.5	Automatic composition of AXMEDIS objects .....	25
3.1.6	Automatic formatting of AXMEDIS objects .....	25
3.1.7	Automatic publication of AXMEDIS objects on AXEPTool.....	26
3.1.8	Automatic programme and publication of AXMEDIS objects on distribution channels .....	27
3.1.9	Acquisition of AXMEDIS objects from the distributor.....	28
3.1.10	Viewing/Using of AXMEDIS objects .....	29
<b>4</b>	<b>AXMEDIS OBJECT EDITING .....</b>	<b>30</b>
4.1	AXMEDIS EDITORS, AS AUTHORING TOOLS.....	30
4.1.1	Creation of a new AXMEDIS object.....	30
4.1.2	Load and save AXMEDIS objects.....	30
4.1.3	Navigating through AXMEDIS objects .....	31
4.1.4	Adding AXMEDIS elements to an existing AXMEDIS object.....	32
4.1.5	Extracting AXMEDIS elements .....	33
4.1.6	Removing an element from an AXMEDIS Object .....	33
4.1.7	Moving an element within the AXMEDIS Object.....	34
4.1.8	Adding a resource .....	34
4.1.9	Managing/Modifying a resources .....	35
4.1.10	Navigating and understanding DRM rules and PAR.....	36
4.2	AXMEDIS INTERNAL VIEWERS .....	37
4.2.1	Invoking an internal viewer/editor .....	37
4.2.2	Managing a digital resource by respecting the DRM in an Internal Viewer/Editor.....	37
4.2.3	Closing an Internal viewer/editor .....	38
4.3	AXMEDIS VISUAL AND BEHAVIOURAL EDITOR.....	39
4.3.1	Editing the visual scene for SMIL scene.....	39
4.3.1.1	Creating and deleting an Element in a SMIL scene .....	39
4.3.1.2	Resizing and moving the Element in a SMIL scene .....	40
4.3.1.3	Changing the background color of the visual Element .....	40
4.3.1.4	Association of media resources with an Element of a SMIL Scene .....	41
4.3.2	Editing the temporal information of media resources.....	41
4.3.2.1	Editing the unit and length of timeline .....	41
4.3.2.2	Editing the displaying time boundary of each media resource .....	42
4.3.3	Previewing the SMIL Scene after editing.....	43
4.3.4	Loading and saving the SMIL component into AXMEDIS object.....	43
4.4	NAVIGATION AND HYPERLINKING WITH MULTIPLE SMIL SCENES .....	44
4.5	AXMEDIS OBJECT EDITOR AND VIEWERS .....	44
4.5.1	Opening annotations and comments of the media object.....	44
4.5.2	Adding annotations and comments of the media object.....	45
4.5.2.1	Adding audio annotations and comments of the media object .....	45
4.5.2.2	Adding text annotations and comments of the media object .....	45
4.5.2.3	Adding graphical annotations and comments of the media object .....	46
4.5.3	Saving annotations and comments of the media object .....	46
4.5.4	Removing annotations and comments of the media object.....	46

4.6	AXMEDIS TOOLS FOR USING/PRODUCING AXMEDIS OBJECTS IN OTHER CONTENT TOOLS.....	47
4.6.1	Invoking an external tool with a digital resource belonging to the AXMEDIS object.....	47
4.6.2	Managing the digital resource by respecting the DRM in an external tool.....	48
4.6.3	Closing an External Tool.....	48
4.6.4	Updating a digital resource modified by an external tool.....	49
4.6.5	Transferring a digital resource to an external tool.....	49
<b>5</b>	<b>AXMEDIS PLUG IN DEFINITION.....</b>	<b>51</b>
5.1.1.1	Defining a AXCP plugin.....	51
<b>6</b>	<b>AXMEDIS PRODUCTION TOOLS.....</b>	<b>52</b>
6.1	AUTOMATIC PRODUCTION TOOLS.....	52
6.1.1	AXMEDIS Content Processing Engine.....	52
6.1.1.1	Firing an AXCP rule.....	52
6.1.1.2	Searching for a rule Executor.....	52
6.1.1.3	Automatic production.....	52
6.1.1.4	Verification of the compatibility of DRM associated with digital resources.....	53
6.1.1.5	Verification of rights for digital resources.....	54
6.1.1.6	Embedding a digital resource in the new AXMEDIS object.....	54
6.1.1.7	New AXMEDIS objects generation.....	54
6.1.1.8	Fingerprint estimation of a digital resource.....	55
6.1.1.9	Formatting of AXMEDIS Objects.....	55
6.1.1.10	Adaptation of a digital resource.....	56
6.1.1.11	Protection of the new AXMEDIS object.....	56
6.1.1.12	Merging component's DRM/PAR rules into a new AXMEDIS object.....	57
6.1.1.13	External Tools execute formatting operations.....	57
6.1.2	AXCP Rules Editor.....	58
6.1.2.1	Create a new AXCP rule.....	58
6.1.2.2	Search and Select an AXCP rule.....	58
6.1.2.3	Activating an AXCP rule.....	59
6.1.2.4	Debugging/simulation of an AXCP rule.....	59
6.2	FORMATTING TOOLS.....	60
6.2.1	Automatic Formatting Tools.....	60
6.2.1.1	Automatic formatting process.....	60
6.2.1.2	Automatic template selection.....	60
6.2.1.3	Automatic style-sheet selection.....	61
6.2.1.4	Automatic style-sheet optimization.....	61
6.2.1.5	Format creation.....	62
6.2.2	Interactive Formatting Tools.....	62
6.2.2.1	Interactive formatting process.....	62
6.2.2.2	Interactive template selection.....	62
6.2.2.3	Interactive style-sheet selection.....	63
6.2.2.4	Interactive style-sheet optimization.....	63
6.2.2.5	Template creation.....	64
6.2.2.6	Style-sheet creation.....	64
<b>7</b>	<b>AXMEDIS WORKFLOW.....</b>	<b>66</b>
7.1	WORKFLOW SCENARIOS.....	66
7.2	CONTROLLING AND SUPERVISING LOCAL AXMEDIS TOOLS.....	72
7.2.1	General WorkFlow Use Cases.....	72
7.2.1.1	Create NPD Workspace.....	72
7.2.1.2	Add components to the NPD.....	72
7.2.1.3	Edit information of the NPD.....	73
7.2.1.4	Delete information of a NPD.....	73
7.2.1.5	Show information regarding components of a NPD.....	74
7.2.1.6	Delete a NPD.....	74
7.2.1.7	Search a NPD.....	74
7.2.1.8	Track Component.....	75
7.2.1.9	Timestamp Generator.....	75
7.2.1.10	List of Work.....	76
7.2.1.11	Select a Work Item from the List of Work.....	76
7.2.1.12	Complete a task of a work Item.....	76
7.2.1.13	Change State/Phase of a Task for a work Item.....	77
7.2.1.14	Notification of information to a personnel for a task of a work.....	77
7.2.1.15	Global Viewer of all information of a NPD.....	78
7.2.1.16	Check-in task performed by manual operator.....	78

7.2.1.17	Check-out task performed by manual operator .....	78
7.3	CONTROLLING AND SUPERVISING AXMEDIS OBJECT LIFE IN AXMEDIS COMPLIANT FACTORIES .....	79
<b>8</b>	<b>AXMEDIS OBJECT ACQUISITION FROM CMS.....</b>	<b>81</b>
8.1	AUTOMATIC GATHERING OF CONTENT, COLLECTOR ENGINE .....	81
8.1.1	Setup for metadata mapping .....	81
8.1.2	Setup for content crawling.....	81
8.1.3	Define what content to acquire from Crawled Integrated Database.....	82
8.1.4	Start content crawling .....	82
8.2	FINGERPRINT EXTRACTOR AS A COLLECTION OF COLLECTOR ENGINE PLUG-INS FOR EXTRACTING FEATURES 83	
8.2.1	Calculating content descriptors/fingerprint during crawling .....	83
<b>9</b>	<b>AXMEDIS DATABASE .....</b>	<b>85</b>
9.1	MANAGING A DATABASE OF AXMEDIS OBJECTS .....	85
9.1.1	Administer Objects in the AXMEDIS DB .....	85
9.1.2	Administer User in the AXMEDIS DB .....	85
9.1.3	Accessing a specific version of an AXMEDIS object .....	85
9.1.4	Removing last version of an AXMEDIS object.....	86
9.1.5	Removing an AXMEDIS object.....	86
9.1.6	User Management .....	87
9.1.7	User Groups Management .....	87
9.2	MAKING QUERIES INSIDE DATABASES OF AXMEDIS OBJECTS AND INSIDE THE OBJECTS .....	87
9.2.1	Querying for AXMEDIS objects and inside objects.....	88
9.2.2	Querying for AXMEDIS from Clients .....	89
9.2.3	Bookmark a query.....	90
9.2.4	Retrieve a bookmarked query.....	90
9.2.5	Organize bookmarked queries .....	91
9.2.6	Save an incomplete query .....	91
9.2.7	Retrieve an incomplete query .....	92
<b>10</b>	<b>AXMEDIS AXEPTOOLS FOR P2P DISTRIBUTION ON B2B.....</b>	<b>93</b>
10.1	AXEPTOOL FOR P2P ON B2B.....	93
10.1.1	Discovery and connection of peers on B2B P2P network.....	93
10.1.2	Report P2P downloads/uploads network traffic .....	93
10.2	PUBLICATION AND LOADING AXMEDIS OBJECTS OF AXEPTOOL .....	93
10.2.1	Creation of a publishing rule for the AXEPTool.....	94
10.2.2	Automatic publication of a selection of objects on the AXEPTool .....	94
10.2.3	Automatic updating of a modified object on the AXEPTool .....	95
10.2.4	Manual Publication of AXMEDIS Objects with the AXEPTool.....	96
10.2.5	Producing a query to search on the AXEPTool network.....	96
10.2.6	View/Manage query results coming from the AXEPTool .....	97
10.2.7	Active query pool management for the AXEPTool .....	97
10.2.8	Downloading an AXMEDIS object .....	97
10.2.9	Automatic downloading of a selection of objects available in the P2P network.....	98
10.2.10	Selecting objects for the AXDB from the those downloaded .....	99
10.2.11	Automatic loading new versions of AXMEDIS Objects for the AXEPTool .....	100
10.2.12	Automatic loading new AXMEDIS Objects with the AXEPTool.....	101
10.2.13	Manual Loading of AXMEDIS Objects with the AXEPTool.....	101
10.2.14	Creation of a loading rule for the AXEPTool.....	102
10.2.15	Preview an AXMEDIS object content coming from AXEPTool.....	102
<b>11</b>	<b>PROGRAMME AND PUBLICATION ENGINE TOOLS .....</b>	<b>104</b>
11.1	PROGRAMME AND PUBLICATION PROGRAMME PRODUCTION .....	104
11.2	PROGRAMME AND PUBLICATION PROGRAMME EDITING.....	104
11.3	ACTIVATION OF PROGRAMME AND PUBLICATION PROGRAMMES .....	105
11.4	LAUNCH OF PROGRAMME AND PUBLICATION PROGRAMMES FROM WORKFLOW .....	106
11.5	TRIAL PRE-ACTIVATION OF PROGRAMME AND PUBLICATION PROGRAMME.....	108
11.6	MONITORING OF PROGRAMME AND PUBLICATION ENGINE.....	108
<b>12</b>	<b>AXMEDIS AXEPTOOLS FOR SATELLITE DATA BROADCAST ON B2B .....</b>	<b>110</b>
12.1	AXMEDIS B2B CLIENT APPLICATION .....	110

12.1.1	B2B Client Installation .....	110
12.1.2	B2B Client Customization .....	110
12.1.3	B2B Client Registration .....	111
12.2	ENABLING A B2B RECEIVING STATION .....	111
12.3	DOWNLOADING AXMEDIS OBJECTS FROM AXEPTOOL BY USING SATELLITE DATA BROADCAST ON B2B .....	112
12.3.1	Pushing an AXMEDIS Object by B2B Carousel .....	112
12.4	AUTOMATIC CONTENT RECEPTION VIA SATELLITE .....	113
12.5	CONTENT DELIVERY VIA SATELLITE .....	113
12.6	CONTENT PROTECTION FOR SATELLITE DISTRIBUTION .....	114
<b>13</b>	<b>AXMEDIS PROTECTION TOOLS .....</b>	<b>115</b>
13.1	SUPER AXCS .....	115
13.1.1	AXMEDIS Registration of AXCSs .....	115
13.1.2	Tool/device off-line registration .....	115
13.1.3	AXMEDIS Object ID Generation .....	115
13.1.3.1	Generation of unique Object ID .....	116
13.1.3.2	Registration of metadata about a new object .....	116
13.1.4	Global Object List WEB Service .....	117
13.1.4.1	Search of AXMEDIS Objects .....	117
13.1.5	Super AXCS Collector .....	117
13.1.5.1	On-line transfer among AXCSs and SuperAXCSs .....	117
13.1.5.2	Off-line synchronization among AXCSs and SuperAXCSs .....	118
13.2	AXMEDIS CERTIFIER AND SUPERVISOR .....	118
13.2.1	AXMEDIS Registration Service .....	118
13.2.1.1	End User registration in a distribution channel .....	118
13.2.1.2	End User registration in AXMEDIS system without a referencing distributor .....	119
13.2.1.3	End User registration in a different distribution channel .....	120
13.2.1.4	Registration of a new structured group of people .....	121
13.2.1.5	Registration of an old User of the Channel on AXMEDIS .....	122
13.2.1.6	User password modification .....	123
13.2.2	AXMEDIS Certification and Verification .....	124
13.2.2.1	Authentication of a Tool and a Device .....	124
13.2.2.2	Certification of AXMEDIS Tool by a User on a Device .....	124
13.2.2.3	Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption .....	127
13.2.2.4	Reverification of AXMEDIS users using AXMEDIS tools on a Device during content consumption inside a domain .....	128
13.2.3	Manual Blocking .....	129
13.2.3.1	Manual User Blocking/Unblocking .....	129
13.2.3.2	Certified Tool blocking/unblocking .....	130
13.2.3.3	Registered Tool blocking/unblocking .....	130
13.2.4	AXMEDIS Supervisor .....	131
13.2.4.1	AXMEDIS Protection Information delivery .....	131
13.2.4.2	Storage/update of protection information of an AXMEDIS Object to the AXCS .....	132
13.2.4.3	Storage of Action Logs in AXCS Accounting database .....	132
13.2.4.4	Storage of SupervisorInputData in AXCS Accounting database .....	132
13.2.5	AXMEDIS Reporting and Statistics Web Service .....	133
13.2.5.1	Object usage reporting for accounting purposes .....	133
13.2.5.2	Object usage reporting for statistics purposes .....	133
13.2.6	Accounting Manager and Reporting Tool .....	134
13.2.6.1	List of all operations performed on an object .....	134
13.2.6.2	List of all operations performed by a user .....	134
13.2.6.3	Usage report about an object .....	134
13.2.6.4	Usage report about a distributor .....	135
13.2.6.5	Usage report about a provider .....	135
13.2.6.6	List objects for which an administrative account can be requested .....	135
13.2.6.7	Listing AXMEDIS clients of a distributor/channel .....	136
13.2.6.8	Listing distributors .....	136
13.3	PROTECTION TOOL ENGINE .....	137
13.3.1	Content protection .....	137
13.3.2	Create a new protection rule .....	138
13.3.3	Search and Select a protection rule .....	139
13.3.4	Activating a protection rule .....	139
13.3.5	Removing a protection rule .....	140
13.3.6	Debugging a protection rule .....	140
13.3.7	Editing protection rules .....	140

13.3.8	Printing protection rules .....	141
13.3.9	Automated creation and association of licences .....	141
13.3.10	Automated verification of licences or PARs .....	142
13.3.11	Automated editing of PARs .....	142
13.3.12	Automated editing of licences.....	143
13.4	ADMINISTRATIVE INFORMATION INTEGRATOR .....	144
13.4.1	Integrating Distributor administrative information of the basis of End User actions.....	144
13.4.2	Integrating Collecting Society administrative information of the basis of End User actions .....	145
13.4.3	Distributor asks for administrative information.....	146
13.4.4	Administrative information retrieval for distributors .....	147
13.4.5	Administrative information retrieval for collecting societies.....	147
13.5	PROTECTION MANAGER SUPPORT/SERVER GENERAL .....	148
13.5.1	Protection Manager Support / Server.....	148
13.5.1.1	Consumption of a protected and governed AXMEDIS object in a connected environment.....	148
13.5.1.2	Consumption of a protected and governed AXMEDIS object in a unconnected environment.....	149
13.5.1.3	Protection of an AXMEDIS object.....	150
13.5.1.4	Protection and association of licenses of/to an AXMEDIS object.....	151
13.5.1.5	Renewal of IPMP information after the detection of a succeed attack (connected).....	151
13.5.2	DRM Support .....	152
13.5.2.1	License creation for new content .....	152
13.5.2.2	License creation for cross-media content .....	152
13.5.2.3	License verification against parent licenses .....	153
13.5.2.4	License verification against PAR.....	153
13.5.2.5	User authorisation on unconnected environment .....	154
13.5.2.6	User authorisation on semiconnected environment (PMS server Online, AXCS offline) .....	154
13.5.2.7	User authorisation on fully connected environment (PMS server Online, AXCS online).....	155
13.5.2.8	Navigation of licensing information .....	156
13.5.2.9	Rights Expression Translator .....	156
13.5.2.10	License migration .....	156
13.5.2.11	Cooperative Authorization Check.....	157
13.6	ENCRYPTION/DECRYPTION SUPPORT.....	157
13.6.1	Encryption.....	157
13.6.2	Decryption .....	158
13.6.3	Encryption of symmetric key .....	158
13.6.4	Decryption of symmetric key .....	158
<b>14</b>	<b>AXMEDIS PLAYER.....</b>	<b>159</b>
14.1	AXMEDIS PLAYER ON PC, TABLET PC .....	159
14.1.1	Content Recording for Playtime Shift.....	159
14.1.2	Fast-forward of Content in Internal Players/Viewers.....	160
14.1.3	Local adaptation of Content in Internal Players/Viewers.....	160
14.1.4	Annotate for personal use.....	161
14.1.5	Local User Profiles .....	161
14.1.6	History of the last played contents .....	162
<b>15</b>	<b>AXMEDIS FOR DISTRIBUTION VIA INTERNET .....</b>	<b>163</b>
15.1	BACK OFFICE MANAGEMENT .....	163
15.1.1	Creating a New Mediaclub.....	163
15.1.2	Mediaclub Setup.....	163
15.1.3	Mediaclub Accounts and Permission Management.....	164
15.1.4	Mediaclub Project Uploading and publishing contents.....	164
15.1.5	Mediaclub Project Acquiring AXMEDIS content.....	164
15.1.6	Mediaclub Project define payment gateway entry.....	165
15.1.7	Mediaclub Shop payment Management.....	165
15.1.8	Mediaclub Shop Management refund a transaction .....	166
15.2	END USER CLIENT CONFIGURATION .....	166
15.2.1	User Software Installation .....	166
15.2.2	User Registration .....	166
15.3	USER LOGIN.....	167
15.3.1	Authentication trough AXMEDIS client .....	167
15.3.2	Authentication trough an external SSO system .....	167
15.4	CATALOGUE BROWSING .....	168
15.4.1	Catalogue Listing.....	168

15.4.2	Catalogue Searching.....	169
15.4.3	Available resources listing .....	169
15.4.4	Content Access .....	170
15.4.5	User Page .....	170
15.5	CATALOGUE CONTENT PURCHASE .....	171
15.5.1	Content Fetching.....	171
15.5.2	User Authentication Form .....	171
15.5.3	Catalogue Content Transaction .....	172
15.5.4	Content Access .....	173
15.5.5	Content Preview .....	173
15.5.6	License Acquisition .....	173
15.5.7	Multi-device license activation and back-up .....	174
15.5.8	Pre-ordering and registration for a group of students.....	174
15.6	BUSINESS MODELS.....	175
15.6.1	Rental.....	175
15.6.2	pay per download.....	176
15.6.3	Sell-through .....	176
15.6.4	subscription.....	176
15.6.5	pay per minute .....	177
15.6.6	pay per Kb downloaded.....	177
15.6.7	pay per day.....	178
15.6.8	pay per credits.....	178
15.6.9	Grouped licenses.....	178
15.6.10	Packaged offers.....	179
15.7	ADVANCED PAYMENT METHODS .....	179
15.7.1	Gift Certificates .....	179
15.7.2	Wallet.....	181
<b>16</b>	<b>AXMEDIS FOR DISTRIBUTION TOWARDS MOBILES .....</b>	<b>182</b>
16.1	GENERAL ASSUMPTIONS AND NOTES TO ARCHITECTURE.....	182
16.2	USE CASES .....	182
16.2.1	Domain registration .....	182
16.2.2	Content Preparation/ingestion.....	183
16.2.3	Content Retrieving Criteria Management.....	183
16.2.4	Content Retrieving Criteria Definition .....	183
16.2.5	Content Retrieving Criteria Selection.....	184
16.2.6	Content Retrieving Criteria Removing .....	184
16.2.7	Supported device profile adding .....	185
16.2.8	Supported device profile removing.....	185
16.2.9	User registration by administrator.....	186
16.2.10	User update by administrator .....	186
16.2.11	User remove by administrator .....	187
16.2.12	User roles management .....	187
16.2.13	User registration .....	188
16.2.14	Certification of users .....	188
16.2.15	Client application download.....	189
16.2.16	User login.....	189
16.2.17	User interface language selection.....	190
16.2.18	Catalogue loading and browsing.....	190
16.2.19	Contents Search .....	191
16.2.20	Getting content information .....	192
16.2.21	Content Preview .....	192
16.2.22	Content Delivery.....	192
16.2.23	Content Acquire.....	193
16.2.24	Content fruition.....	194
16.2.25	User Data Update.....	194
<b>17</b>	<b>AXMEDIS FOR DISTRIBUTION TOWARDS I-TV.....</b>	<b>196</b>
17.1	USER TERMINAL INSTALLATION AND CONFIGURATION.....	196
17.1.1.1	PC+DVB Card Terminal.....	196
17.1.2	User Software Installation .....	196

17.1.3	User Registration .....	197
17.1.3.1	Application Selection .....	197
17.1.3.2	User Profiling .....	198
17.2	CONTENT LISTING .....	198
17.2.1	Content Web Listing .....	198
17.2.2	Content Carousel Listing .....	199
17.3	CONTENT VOTING .....	200
17.4	CONTENT SELECTION .....	200
17.4.1	Manual Content Selection .....	200
17.4.2	Automatic Content Selection .....	201
17.5	CONTENT RECEPTION .....	201
17.6	CONTENT REPARATION .....	201
17.7	CONTENT ACCESS .....	202
17.8	CONTENT PREVIEW .....	202
17.9	LICENSE ACQUISITION .....	203
17.9.1	User Identification .....	204
17.9.2	Billing .....	204
17.10	CONTENT BACKUP .....	205
17.11	CONTENT RESTORE .....	205
17.11.1	Cache Preloading .....	206
17.12	CACHE CLEANING .....	206
17.13	CACHE-BASED PERSONALISED CONTENT DISTRIBUTION SPECIFIC USE CASES .....	207
17.13.1	Automatic Content Access Set Up .....	207
17.13.2	AXMEDIS Channel personalisation .....	207
17.13.3	Automatic Content Access .....	207
17.13.4	AXMEDIS Channel PVR functionalities .....	208
<b>18</b>	<b>AXMEDIS FOR DISTRIBUTION TO PDA VIA KIOSKS .....</b>	<b>209</b>
18.1	CONTENT CATALOGUE CREATION .....	209
18.2	CONTENT CATALOGUE LOADING (PUBLICATION) .....	209
18.3	CONTENT CATALOGUE LOADING .....	210
18.4	USER REGISTRATION TO KIOSK .....	210
18.5	USER LOGIN .....	211
18.6	CONTENT BROWSING & PREVIEWING .....	212
18.7	CONTENT SELECTION AND CHART MANAGEMENT .....	213
18.8	CHECK OUT PROCEDURE INITIATION .....	213
18.9	PURCHASING / ACQUIRING / RENTING .....	214
18.10	REPOSITORY SELECTION .....	214
18.11	DESTINATION TARGET IDENTIFICATION (UNIQUE ID FOR TARGET – WIFI) .....	215
18.12	DELIVERY TEMPLATE SELECTION (DEPENDING ON DEVICE) .....	215
18.13	DELIVERY FORMAT SELECTION (DEPENDING ON CONTENT) .....	216
18.14	BILLING .....	216
18.15	DATA DELIVERY .....	216
18.16	CHECK OUT PROCEDURE CLOSURE .....	217
18.17	SUCCESSFUL DELIVERY CHECK (RECOVERY IN CASE OF FAILURE) .....	217
18.18	CONTENT FRUITION AFTER DOWNLOAD ON PDA OR MOBILE .....	220
18.19	USER PROFILE CHANGE .....	220
18.20	USER DEVICE CONFIGURATION & APPLICATION FRONT-END INSTALLATION .....	221
18.21	CONTENT UPDATE (VIA SATELLITE) .....	222
<b>19</b>	<b>COMPOSITE USE CASE: AUTOMATIC CONTENT PRODUCTION .....</b>	<b>223</b>
<b>20</b>	<b>COMPOSITE USE CASE: CONTENT EDITING, PROTECTION AND AUTHORING .....</b>	<b>224</b>
<b>21</b>	<b>COMPOSITE USE CASE: CONTENT STORAGE AND SEARCH, DATABASE MANAGEMENT ..</b>	<b>225</b>
<b>22</b>	<b>COMPOSITE USE CASE: CONTENT ACQUISITION FOR THE DOMAIN AND USAGE IN THE DOMAIN .....</b>	<b>226</b>
<b>23</b>	<b>COMPOSITE USE CASE: CONTENT POSTING IN THE P2P AND SEARCH FROM THE P2P .....</b>	<b>226</b>

24	COMPOSITE USE CASE: PROGRAMME PRODUCTION AND PUBLICATION.....	227
25	COMPOSITE USE CASE: WORKFLOW PROCESS DEFINITION .....	228
26	COMPOSITE USE CASE: CONTENT BEHAVIOUR DEFINITION AND USAGE ON THE CLIENT SIDE	230
27	COMPOSITE USE CASE: PRODUCTION AND EXECUTION OF CONTENT PROCESSING RULES	230
28	COMPOSITE USE CASE: CONTENT PRODUCTION AND USAGE .....	231
29	COMPOSITE USE CASE: CONTRACT READING AND DIGITAL LICENSE PRODUCTION AND USAGE .....	232
30	COMPOSITE USE CASE: ACCESS TO ACTION LOGS AND THEIR USAGE.....	233

## 1 Executive Summary and Report Scope

Market and end-users are pressing content industry to reduce prices. This is presently the only solution to setup viable and sustainable business activities with e-content. Production costs have to be drastically reduced while maintaining product quality. Content providers, aggregators and distributors need innovative instruments to increase efficiency. A solution is automating, accelerating and restructuring the production process to make it faster and cheaper. The goals will be reached by: (i) accelerating and reducing costs for content production with artificial intelligence algorithms for content composition, formatting and workflow, (ii) reducing distribution and aggregation costs, increasing accessibility, with a P2P platform at B2B level integrating content management systems and workflows, (iii) providing algorithms and tools for innovative and flexible Digital Rights Management, exploiting MPEG-21 and overcoming its limits, supporting several business and transactions models.

AXMEDIS consortium (producers, aggregators, distributors and researcher) will create the AXMEDIS framework with innovative methods and tools to speed up and optimise content production and distribution, for *production-on-demand*. The content model and manipulation will exploit and expand MPEG-4, MPEG-7 and MPEG-21 and others real and de-facto standards.

AXMEDIS will realize demonstrators, validated by means of real activities with end-user by leading distributor partners: (i) tools for content production and B2B distribution; (ii) content production and distribution for i-TV-PC, PC, kiosks, mobiles, PDAs. The most relevant result will be to transform the demonstrators into sustainable business models for products and services during the last project year. Additional demonstrators will be some associated projects launched as take up actions. The project will be supported by activities of training, management, assessment and evaluation, dissemination and demonstration at conference and fairs.

This deliverable is a revised version of the early use cases it is related to all the deliverables of WP2 which is devoted to the continuous collection and analysis of user requirements. This activity is performed by setting up a user group of experts and by considering the content production models, educational paradigms, entertainment models, distribution paradigms and protection innovative aspects of the project. The work includes the adoption of interviews and the identification of use cases, description of the test cases, (while the corresponding collection of reference content for stressing key problems and for the eventual verification and validation of corresponding solutions is performed in WP8), collection of current practices (best practices) in using media technologies and solutions (processes, tools, methodologies, equipment, etc), identification of distribution processes and models.

Main deliverables are:

- DE2.3.1.2 – User Group Maintenance (M13) -- responsible UNIVLEEDS -- report on the activity related in the management and improvement of the user group, enlargement of it, analysis of the coverage of the UG with respect to the project topics, etc., activity to be performed in the next months (evolution of DE2.3.1);
- DE2.1.1.2.1 -- User Requirements, first update (M16) – responsible DSI -- this deliverable contains the revised and updated version of the user requirement produced in DE2.1.1; This deliverable has been planned since the Annex I and it has been split in the following two deliverables that initially have been planned to be under the same number. They were too large document to be considered single documents in this phase;
- DE2.1.1.2.2 – Use Cases and Scenarios, first update (M17) – responsible DSI -- this deliverable contains the revised and updated version of the Use cases and scenarios produced in DE2.1.1;
- DE2.2.1.2 – Test cases and content description, first update (M20) – responsible FUPF -- this deliverable contains the revised and updated version of the test cases for research functionalities and AXMEDIS tool validation, starting from the DE2.2.1; In this case, the description of test cases will be more precise since the first results coming from the WP8 will be available and thus effective links from what can be done for testing and which content has to be used will be possible.

- DE2.3.1.3 – User Group Maintenance, first update (M20) – responsible UNIVLEEDS -- report on the activity related in the management and improvement of the user group, its enlargement, analysis of the coverage of the UG with respect to the project topics, etc., activity to be performed in the next months (evolution of DE2.3.1.2);

The main activities that have supported the production of this deliverable are related to:

**WP2.3 -- Set up and management of a AXMEDIS User Group** -- responsibility UNIVLEEDS -- the established user group of experts will be enlarged and kept informed. The members will receive updated news about project evolution and will constitute a source for requirements and use cases collection and validation; moreover they will contribute to testing and validating produced results. The user group presently presents several experts representing the different users of AXMEDIS tools both at business and consumer levels. They include content producers, content integrators, content designers, usability experts, content distributors, content aggregators, publishers, etc.

**WP2.4 -- Updating requirements analysis after first period** -- responsibility DSI – (M16-M17) In this WP, the updating of the requirements and of use cases and test cases in the periods after the first is performed. Updating means to revise requirements, use cases and test cases, in order to see if they need to be revised and/or improved and/or deleted and/or added according to changes in the state of the art, needs of the context and users, etc. This process of updating is continuously performed and after each intermediate validation such as that of M14 and during the final validation. Requirements, use cases and test cases will be updated by considering the points of view of content designers, multimedia producers, integrators, final users, taking into account project partners, their experts, and experts of the user group by using specific interview based on guidelines produced by the consortium. Other sources of information will be the monitoring and participation to MPEG-21, DMP, and other groups. The use cases and test cases will be structured according to UML methodology creating and updated also scenarios as performed in the first period. As a result a new version of related deliverables will be produced updating and expanding those collected and reported in the deliverables of the first period. The test cases will be used for validating the functionalities identified by research and development WPs and during the activities of integration and optimisation, and in those of demonstration which is temporally allocated after the M30. In this case, the test cases will be more precise since the first results coming from the WP8 will be available and thus effective links from what can be done for testing and which content has to be used will be possible. The Content for the test cases will be collected and/or produced in WP8. The test cases will be structured according to structure of the AXMEDIS framework and tools as defined and developed in the first 12 months of the project.

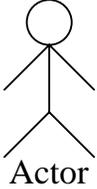
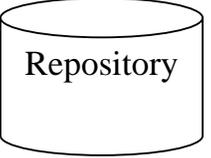
Test cases are reported in a different deliverable.

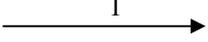
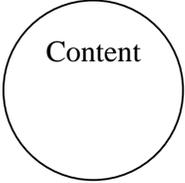
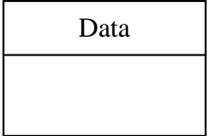
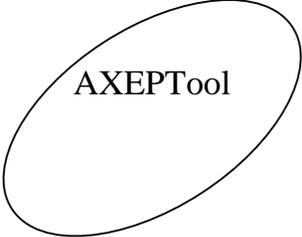
## 2 Structure of Use Cases

### 2.1 Structure of Use Cases

<b>UCId</b>	Unique identifier of the use case
<b>Use case</b>	Name of the use case
<b>Description</b>	Plain description of the use case activation, execution and termination
<b>Actors</b>	People, tools or entities involved in the use case, e.g. who (or what) activates the use case
<b>Assumptions</b>	Conditions which must be satisfied before use case activation
<b>Steps</b>	Step by step description of the use case activation, execution and termination
<b>Post-conditions</b>	Conditions which must be satisfied after use case termination
<b>Variations</b>	Use case variations which could be relevant by the end-user point of view and that are similar to the main use case for developers
<b>Asynchronous actions</b>	Important actors' actions which change standard use case step flow, e.g. during a background search an actor could stop it by clicking on the stop button. For each asynchronous actions, relevant post-conditions should be reported
<b>Design suggestions</b>	Useful hints or implications about the thought project structure regarding the use case
<b>Issues</b>	Possible issues, notes or annotations related to the use case implementation

### 2.2 Use Case and Scenario diagram: shapes and semantics

Shape	Name	Semantic
 <p>Actor</p>	Actor	The shape represents one of the Actor declared in the related use case or scenario
 <p>Tool</p>	Tool	The shape represents the tool whose name (defined in the specifications document) is contained within the shape, e.g. an engine or the AXMEDIS Editor, etc...
 <p>Support</p>	Support/library	The shape represents the support whose name (defined in the specifications document) is contained within the shape, e.g. AXDBM, etc...
 <p>Repository</p>	Repository	The shape represents a data repository whose name (defined in the specifications document) is contained within the shape, e.g. the local AXDB or the "Repository of Publication Rules/Selections" in "Programme and Publication Area"

	<p>Interaction</p>	<p>The shape represents an interaction between two modules, e.g. a call to an available function in another module, a data flow between tools, etc...</p> <p>The number above the arrow should correspond to a step of the related use case or scenario. If one step implies more than one interaction among actors, tools and supports than the label could be the number of the step plus a letter, e.g. 1a, 1b, etc...</p> <p>Notice that interaction arrow can not be bidirectional, i.e. if an interaction between two modules implies a double exchange of data you will draw two one-way interactions.</p> <p>In use cases and scenarios description, you should explain what kind of interaction a step implies, i.e. a data or content transmission, function call, use, etc...</p>
	<p>Unprotected AXMEDIS Object or Content</p>	<p>The shape represents a raw content (mp3, wav, etc...) or an unprotected AXMEDIS object. The shape could contains the "name" of content with respect to the related use case</p>
	<p>Protected AXMEDIS Object or content</p>	<p>The shape represents a protected AXMEDIS object. The shape could contains the "name" of content with respect to the related use case</p>
	<p>Data</p>	<p>The shape represents data, other than content and protected/unprotected AXMEDIS object, used in the use case, e.g. selection, rule, etc...</p>
	<p>AXEPTool Distributed Database</p>	<p>The shape represents the AXEPTool in its meaning of distributed database as it is used in all other documents of AXMEDIS</p>

### 3 General use cases

The actors used in all the use cases are listed below in alphabetical order:

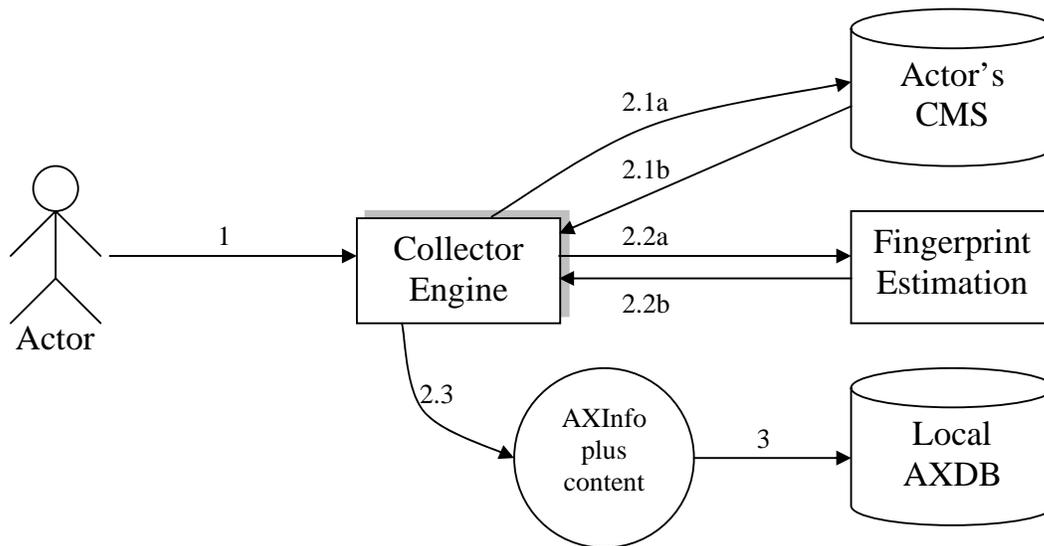
Active Engine	Producer
Aggregator	Programme producer
All AXMEDIS User	Programme manager
Any kind of user	Project Manager
APS (ILabs IRC Application server)	Publisher
AXEPTOOL	Reseller
AXEPTOOL User	Retailer
AXMEDIS B2B Client Application	Shop manager
AXMEDIS B2B Distributor	Subscriber
AXMEDIS Certifier	System
AXMEDIS DB in house	System Manager
AXMEDIS Distributor	The teacher
AXMEDIS Framework	The students
AXMEDIS Node	Transcoding plug-in
AXMEDIS Plug-in	Transcoding server
AXMEDIS Professional user	User device (PDA/Mobile)
AXMEDIS Publication environment	Virtual Learning Environment Provider
AXMEDIS Query Support	
AXMEDIS Synchroniser	
AXMEDIS Workflow Manager	
AXOM	
Collecting societies	
Composer	
Compositional engine	
Content Consumer	
Content Distributor	
Content Integrator	
Content Owner	
Content Provider	
Customer	
Creator	
Designer	
Distributor	
Domain PMS	
Editor	
End user	
End user fruition device	
Fingerprint estimation tool	
Formatting engine	
HME (ILabs IRC Handset management engine)	
Integrator	
Kiosk Authentication Application	
Kiosk front-end	
Kiosk Manager	
Local AXDB	
Local system	
Other application charged of applying actions on AXMEDIS Objects	
PE (ILabs IRC Personalisation engine)	
Performer	

### 3.1 Macro-functionalities

- Automatic collection of content into local AXMEDIS Database from proprietary CMS
- Querying for AXMEDIS objects and Selection creation
- Automatic load (and update) of AXMEDIS objects into local AXDB from AXEPTool
- Automatic protection of AXMEDIS objects
- Automatic composition of AXMEDIS objects
- Automatic formatting of AXMEDIS objects
- Automatic publication of AXMEDIS objects on AXEPTool
- Automatic programme and publication of AXMEDIS objects on distribution channels
- Acquisition of AXMEDIS objects from the distributor
- Viewing/Using of AXMEDIS objects

#### 3.1.1 Automatic collection of content into local AXMEDIS Database from proprietary CMS

A fundamental behaviour of AXMEDIS project is that AXMEDIS will not substitute actually used proprietary CMSs. AXMEDIS will collect contents from those CMSs within content owner's local AXMEDIS Database. Therefore AXMEDIS shall provide an almost automatic way to collect contents from the legacy CMSs.



<b>UCId</b>	UC3.1.1
<b>Use case</b>	Automatic collection of content into local AXMEDIS Database from proprietary CMS
<b>Description</b>	An Actor wants to import content from his/her CMS to his/her local AXMEDIS Database.
<b>Actors</b>	Creator, Producer
<b>Assumptions</b>	The Actor has his/her content digitally stored on a CMS which could be a specifically designed programme, a database or more simply a set of files on a file-system.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using Collector Engine, chooses which content shall be collected. The Actor could specify: <ul style="list-style-type: none"> <li>○ which content have to be collected</li> <li>○ which fingerprints have to be estimated</li> <li>○ mapping of metadata stored in the CMS to the produced objects</li> </ul> </li> <li>2 Collector Engine elaborates the request by: <ol style="list-style-type: none"> <li>2.1 picking up selected content from the Crawler Results Integrated Database</li> <li>2.2 estimating specified fingerprints by using Fingerprint Estimation Tools as Plug-in for AXCP by means of the AXCP Plug-in Manager)</li> <li>2.3 putting together content, fingerprint and all other information needed to fulfil AXInfo schema within a simple AXMEDIS object</li> </ol> </li> <li>3 Collector Engine puts newly created AXMEDIS objects into the local AXDB</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>• The Actor schedules the collection of content triggering it to a specified time and date</li> <li>• The Actor requests to collect content by using the AXMEDIS Workflow Manager</li> <li>• Collector Engine stores the newly created AXMEDIS objects on the local file system, instead of storing it within the local AXDB, depending on the Actor's preferences</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.2 Querying for AXMEDIS objects and Selection creation

Querying and selection of AXMEDIS objects are two of the most used functionalities through AXMEDIS Tools. An example on Selection creation is reported below to better understand it.

#### Example on Selection creation

The Actor executes a series of queries using the AXQS User Interface (which are called Q1, Q2, Q3, Q4 and Q5 below). In that way, he/she can control which AXMEDIS objects satisfy the conditions imposed in the queries. Suppose the Actor receives the following response:

- Q1={AXO1-1, AXO1-2, AXO1-3}
- Q2={AXO2-1, AXO2-2, AXO2-3, AXO2-4, AXO2-5, AXO2-6, AXO2-7, AXO2-8, AXO2-9}
- Q3={AXO3-1, AXO3-2}
- Q4={AXO4-1, AXO4-2, AXO4-3, AXO4-4, AXO4-5}
- Q5={AXO5-1}

where AXOX-X are AXMEDIS object identifiers or something similar.

The Actor wants to create a Selection, he can do that in merging those results in several ways:

- 1) Suppose he/she "likes" (for doing whatever he/she wants) all those objects, than he/she will create Selection S1, e.g. by picking all the check-boxes related to those objects. S1 will be the set of AXMEDIS objects (or object identifiers):

S1={AXO1-1, AXO1-2, AXO1-3, AXO2-1, AXO2-2, AXO2-3, AXO2-4, AXO2-5, AXO2-6, AXO2-7, AXO2-8, AXO2-9, AXO3-1, AXO3-2, AXO4-1, AXO4-2, AXO4-3, AXO4-4, AXO4-5, AXO5-1}

This is yet an expanded Selection because it does not contain Queries.

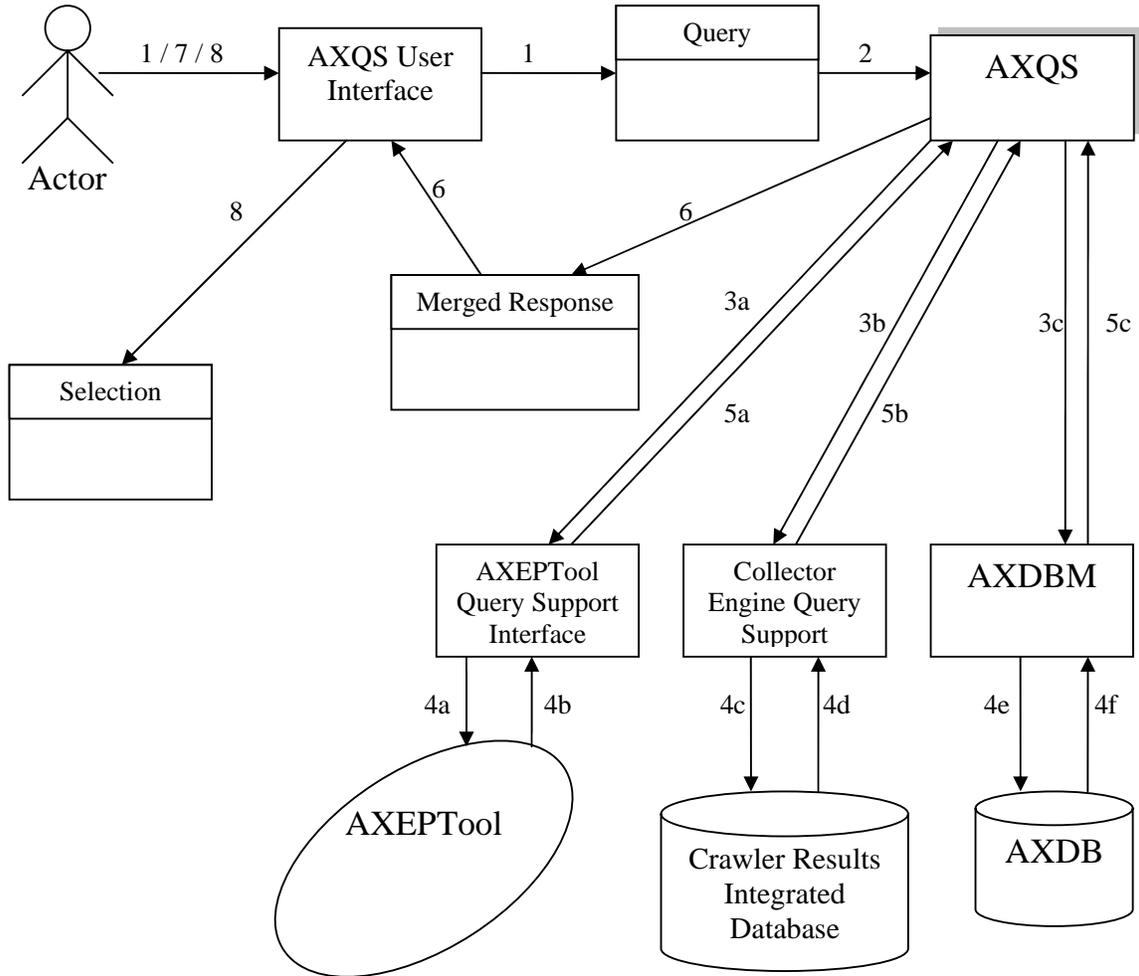
- 2) Suppose he/she "likes" queries Q1, Q2 and Q4, i.e. he/she does not like only the sets of objects obtained at this time but he/she likes the features expressed by the queries. That is, he/she feels he/she will like all objects that can be retrieved at "all time" (not only at this time). Moreover he/she likes objects AXO3-1, AXO3-2 and AXO5-1. He/She will create Selection S2 by picking the check-boxes related to AXO3-1, AXO3-2 and AXO5-1 and those related to Q1, Q2 and Q4 themselves (not related to the objects AXO1-X, AXO2-X and AXO4-X!!!). S2 will be the set of AXMEDIS objects and Queries:

S2={Q1, Q2, Q4, AXO3-1, AXO3-2, AXO5-1}

This is an expandable Selection, i.e. to determine which objects belong to it one shall evaluate the

queries contained in it.

It has to be pointed out that S2 is an “evolving” set of AXMEDIS objects, since the S2 is a non finalised query that may give different answers according to the status of the database. That is, if S2 is expanded approximately at the same time of its creation time the expanded Selection will probably equal to S1, if S2 is expanded a long time after its creation the expanded Selection will probably be a different set of AXMEDIS objects.



<b>UCId</b>	UC3.1.2
<b>Use case</b>	Querying for AXMEDIS objects and Selection creation
<b>Description</b>	An Actor is looking for an AXMEDIS object or a set of AXMEDIS objects which respect a set of technical, right or feature related conditions. The Actor wants to create a Selection on the base of the received responses
<b>Actors</b>	Aggregator, Publisher, Reseller, Retailer
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using the AXQS User Interface, composes a Query on aspects of interest (technical, DRM or feature related). Furthermore, the Actor chooses “where” to search for available AXMEDIS objects: within local AXMEDIS Database (and within AXMEDIS objects contained within the local AXDB), on AXEPTool network or among those contents which have to be collected, by the Collector Engine, and have not yet.</li> <li>2 The Actor submits the queries previously composed</li> <li>3 AXQS submits the Actor’s query to each of the chosen search “places” by using the corresponding specific interface: (i) Collector Engine Query Support Interface, (ii) AXEPTool Query Support Interface and (iii) AXMEDIS Database Manager</li> <li>4 Each query interface (see step 3) looks for the required features in the corresponding domain</li> <li>5 AXQS collects all the responses from the query interfaces</li> <li>6 AXQS merges the results all together and return the complete list to the AXQS User Interface</li> <li>7 AXQS User Interface shows the result to the Actor in an adequate manner, i.e. in such a way that the Actor can understand: (i) from which source an object come (ii) which are the restriction on the object (iii) etc...</li> <li>8 The Actor creates a new Selection (he/she could give it a name or a description)</li> <li>9 The Actor can add to the Selection some of the AXMEDIS object returned (e.g. by picking the checkboxes corresponding to the desired objects) or the query itself (in this case, the Selection could change during the time depending on the evolution of the available objects)</li> <li>10 The Actor can iterate steps 1 to 5 and 7. In that way, the Actor could create a complex Selection which could be composed of several Queries and/or several AXMEDIS objects (e.g. identified by a list of AXOIDS)</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.3 Automatic load (and update) of AXMEDIS objects into local AXDB from AXEPTool

<b>UCId</b>	UC3.1.3
<b>Use case</b>	Automatic load (and update) of AXMEDIS objects into local AXMEDIS Database from AXEPTool
<b>Description</b>	<p>An Actor wants to load within his/her local AXMEDIS Database all AXMEDIS objects (downloaded by the AXEPTool from the P2P Network) which belongs to a given Selection of AXMEDIS objects available on the AXEPTool network. Moreover, the Actor wants:</p> <ul style="list-style-type: none"> <li>• the system will update automatically the previously downloaded objects if they change on the AXEPTool network</li> <li>• the system will load automatically new objects which become element of the given Selection</li> </ul>
<b>Actors</b>	Aggregator, Content Provider, Publisher, etc...
<b>Assumptions</b>	The Actor has previously created a Selection which contains available AXMEDIS objects on the AXEPTool network which satisfy some Actor's needs by using the AXQS User Interface integrated within the AXCP Rule Editor that we can call in this case Publication/Loading Rules/Selections Editor (see use case "Querying for AXMEDIS objects and Selection creation"). In addition, for those object sources, some metadata mappings have been defined by means of the Metadata Mapper. Thus a Rule including the Selection is Activate. This rule is placed in the List of P2P Active Selections/Rules.

<p><b>Steps</b></p>	<ol style="list-style-type: none"> <li>1 Using the Publication/Loading Rules/Selections Editor, the Actor activates the created Selection. Thus a corresponding Rule is created and included in a List of P2P Active Selections/Rules of the Scheduler associated with the AXEPTool P2P Active Selection Engine. This is an instance of the AXCP Engine periodically activating Rules for loading objects from the network. The Actor could also specify:       <ul style="list-style-type: none"> <li>○ an activation period, i.e. a period of time for which the Selection/Rule will be elaborated to update the identified objects</li> </ul> </li> <li>2 AXEPTool P2P Active Selection Engine elaborates the Active Rules:       <ol style="list-style-type: none"> <li>2.1 expanding the Selection towards a query on the P2P network obtaining a list of objects</li> <li>2.2 for each object           <ol style="list-style-type: none"> <li>2.2.1 the AXEPTool downloads the AXMEDIS object from the network</li> <li>2.2.2 the object download is monitored by the AXEPTool Monitor</li> <li>2.2.3 the object is stored on the AXEPTool IN AXMEDIS Database</li> <li>2.2.4 the objects for which the metadata mapping is accessible can be automatically loaded into the AXMEDIS Database, while for the others a alarm has to be generated</li> </ol> </li> </ol> </li> <li>3 when the objects are downloaded, the Actor can try (e.g. play, run, visualize, etc...) each object accordingly to the related DRM rules</li> <li>4 In the case in which point 2.2.4 is not activated and thus the objects are only downloaded from the network and not loaded in the database. After the Actor has tried those objects, he/she can decide more effectively which AXMEDIS object are really interesting for him/her. Then he/she creates another Selection (maybe smaller than the previous one) containing only the interesting AXMEDIS objects, it will be a list of objects IDs</li> <li>5 The Actor activates the new Selection submitting it to the Loading Active Rule Engine. The Actor could also specify       <ul style="list-style-type: none"> <li>○ an activation period, maybe the same as steps 1 and 2</li> </ul> </li> <li>6 The AXEPTool P2P Active Selection Engine       <ol style="list-style-type: none"> <li>6.1 access to the IDs object the object, if needed expanding the selection</li> <li>6.2 for each object which belongs to the expanded Selection           <ol style="list-style-type: none"> <li>6.2.1 apply a metadata mapping to load them into the AXMEDIS object into the local AXDB taking them from the IN database.</li> </ol> </li> </ol> </li> <li>7 Accordingly to the activation period given by the Actor, AXEPTool P2P Active Selection Engine will elaborate the Selections</li> <li>8 If a new version of one of the downloaded object is published on AXEPTool network each node of the P2P is informed and by means of the query it is possible to identify eventual changes and updates:       <ol style="list-style-type: none"> <li>8.1 Publishing and Monitoring Objects is informed of this update by its identical counterpart which belongs to who has published the new version</li> <li>8.2 Publishing and Monitoring Objects alerts AXEPTool P2P Active Selection Engine</li> <li>8.3 AXEPTool P2P Active Selection Engine verifies if the newly published object belongs to one of the active Selections stored in P2P Active Selections</li> <li>8.4 if the object belongs           <ol style="list-style-type: none"> <li>8.4.1 AXEPTool P2P Active Selection Engine will download again the object replacing the old version in the AXEPTool IN AXDB</li> <li>8.4.2 AXEPTool P2P Active Selection Engine will load the object to the local AXDB (supposing the related Selection still active) with a specific mapping of metadata</li> </ol> </li> <li>8.5</li> </ol> </li> </ol>
---------------------	--

<b>Steps</b>	
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Selections can be activated by using the AXMEDIS Workflow Manager that is connected to the AXCP Scheduler</li> <li>• The Actor wants the Selection is elaborated only once, than the Actor specifies the Selection as one-time Selection instead of specifying an activation period</li> <li>• Step 2 can require a large amount of time to be accomplished (due to transfer time and object size). Thus, the Actor should be advised by AXEPTool P2P Active Selection Engine when step 2 has been completed, e.g. via workflow functionalities. A similar observation can be done for step 8</li> <li>• All the above activities can be done asynchronously and in parallel for each selection by independent nodes of the AXCP GRID.</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.4 Automatic protection of AXMEDIS objects

<b>UCId</b>	UC3.1.4
<b>Use case</b>	Automatic protection of AXMEDIS objects
<b>Description</b>	An Actor wants to protect one or some AXMEDIS objects contained within his/her local AXDB (perhaps before distributing them over B2B or B2C networks). Doing that, the Actor wants to add some PAR (Potential Available Rights) to the AXMEDIS objects and he/she also wants to obtain the corresponding licence templates.
<b>Actors</b>	Aggregator, Author, Composer, Performer, Producer, Publisher and Reseller
<b>Assumptions</b>	The Actor has previously created a Selection which contains AXMEDIS objects contained within the local AXDB by using the AXQS User Interface integrated within the AXCP Rule Editor (see use case “Querying for AXMEDIS objects and Selection creation”)

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor composes protection rules by using the AXCP Rules Editor and by using the Protection Information Editor. Such rules specifies:             <ul style="list-style-type: none"> <li>○ the previously created Selection</li> <li>○ PAR to be added</li> <li>○ DRM rules to be added, this is the license that may included or not in the object.</li> <li>○ Protection Information model to be take, how to protect those objects (encryption, specific algorithm, etc...)</li> <li>○ the scheduling for the protection process</li> </ul> </li> <li>2 The Actor, through AXCP Rules Editor, submits the protection rules to the AXCP Rules Engine</li> <li>3 AXCP Rules Engine elaborates the Rules/Selections contained within the Protection Rules/Selections at the set time:             <ol style="list-style-type: none"> <li>3.1 the AXCP Rules Engine the Selection</li> <li>3.2 for each object which belongs to the expanded Selection                 <ol style="list-style-type: none"> <li>3.2.1 AXCP Rules Engine gets the AXMEDIS object by using the AXMEDIS Database Manager</li> <li>3.2.2 AXCP Rules Engine applies all requested protections to the object</li> <li>3.2.3 AXCP Rules Engine adds the required DRM rules and PAR through PMS and AXOM</li> <li>3.2.4 AXCP Rules Engine produces the license template related to PAR added to the object and send them to the License Generator using the PMS</li> <li>3.2.5 In alternative a set of licenses can be created and posted on the PMS server from the Rule.</li> <li>3.2.6 AXCP Rules Engine stores the protected AXMEDIS objects to the local AXDB using AXDB Manager</li> </ol> </li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Protection rules can be activated by using the AXMEDIS Workflow Manager</li> <li>• The Actor could also request the immediate protection of a single AXMEDIS object using the AXCP Rules Engine or by mans of the AXMEDIS Editor with Protection Information Editor.</li> <li>• Protection of content can require a large amount of time to be accomplished (due to content size and applied protections). Thus, the Actor should be advised by AXCP Rules Engine when step 3 has been completed, e.g. via workflow functionalities</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.5 Automatic composition of AXMEDIS objects

<b>UCId</b>	UC3.1.5
<b>Use case</b>	Automatic composition of AXMEDIS objects
<b>Description</b>	An Actor wants to obtain, in an almost automatic way, a set of compounded AXMEDIS objects from a set of AXMEDIS objects
<b>Actors</b>	Aggregator, integrator
<b>Assumptions</b>	The Actor has previously created a set of Selections by using the AXQS User Interface integrated within the AXCP Rules Editor (see use case “Querying for AXMEDIS objects and Selection creation”).
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using AXCP Rules Editor, activates a compositional rule submitting the followings to the AXCP Rules Engine: <ul style="list-style-type: none"> <li>○ a composition rule which could be created on-the-fly or chosen from the Repository of Compositional Rules</li> <li>○ the previously created set of Selections which are the sets of AXMEDIS objects on which the rule will be applied</li> <li>○ the right number and types of parameters (other than Selections) required by the compositional rule signature</li> <li>○ scheduling information</li> <li>○ etc...</li> </ul> </li> <li>2 The AXCP Rules Engine elaborates the activated rules contained into the Internal Scheduler according to the scheduling information. Such elaborations are made in respect of DRM rules of all involved objects.</li> <li>3 AXCP Rule Engine advises the Actor when the composition process is completed using workflow functionalities.</li> </ol>
<b>Post-conditions</b>	All produced objects shall contain a new AXOID and information (contained in AXInfo) such as: their composition, usability, fingerprinting, etc...
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Compositional rules can be activated using the AXMEDIS Workflow Manager</li> <li>• The Actor could also request the composition of a set of AXMEDIS objects by using the AXCP Rules Editor, e.g. to test and verifies the behaviour of a rule</li> <li>• Instead of a rule, the Actor could gives to the AXCP Rule Engine an object template which specifies how to glue together the AXMEDIS objects contained into the Selections</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The meaning of “to elaborate a compositional rule” is explained somewhere else because that also implies to understand how rules are written and which functionalities the compositional rule scripting language provides. See for this the AXMEDIS Production Tools use Cases.

### 3.1.6 Automatic formatting of AXMEDIS objects

<b>UCId</b>	UC3.1.6
<b>Use case</b>	Automatic formatting of AXMEDIS objects
<b>Description</b>	An Actor wants to automatically obtain a formatted AXMEDIS object on the base of rules
<b>Actors</b>	Publisher, Distributor
<b>Assumptions</b>	The Actor has previously created a Selection by using the AXQS User Interface integrated within the AXCP Rules Editor (see use case “Querying for AXMEDIS objects and Selection creation”).

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using AXCP Rules Editor, activates a formatting rule submitting the followings to the AXCP Rule Engine: <ul style="list-style-type: none"> <li>○ a formatting rule which could be created on-the-fly or chosen from the Repository of Rules, the formatting include the SMIL template and a style sheet.</li> <li>○ the previously created Selection which is the set of AXMEDIS objects on which the rule will be applied</li> <li>○ the right number and types of parameters (other than Selection) required by the formatting rule signature</li> <li>○ scheduling information</li> <li>○ etc...</li> </ul> </li> <li>2 The AXCP Rule Engine elaborates the activated rules contained into the Internal Scheduler according to the scheduling information. Such elaborations are made in respect of DRM rules of all involved objects.</li> <li>3 AXCP Rule Engine advises the Actor when the composition process is completed using workflow functionalities.</li> </ol>
<b>Post-conditions</b>	All produced objects shall contain a new AXOID and information (contained in AXInfo) such as: their composition, usability, fingerprinting, etc...
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Formatting rules can be activated using the AXMEDIS Workflow Manager</li> <li>• The Actor could also request the formatting of an AXMEDIS objects by using the AXCP Rules Editor, e.g. to test and verifies the behaviour of a rule</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The meaning of “to elaborate a compositional rule” is explained somewhere else because that also implies to understand how rules are written and which functionalities the compositional rule scripting language provides. See for this the AXMEDIS Production Tools use Cases use Cases.

### 3.1.7 Automatic publication of AXMEDIS objects on AXEPTool

<b>UCId</b>	UC3.1.7
<b>Use case</b>	Automatic publication of AXMEDIS objects on AXEPTool and related P2P Network
<b>Description</b>	An Actor wants to publish his/her AXMEDIS objects on the AXEPTool network to yield them available to the AXMEDIS community
<b>Actors</b>	Aggregator, Producer, etc...
<b>Assumptions</b>	The Actor has previously created a Selection which contains AXMEDIS objects contained within the local AXDB by using the AXQS User Interface integrated within the Publication/Loading Rules/Selections Editor (see use case “Querying for AXMEDIS objects and Selection creation”)

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using Publication/Loading Rules/Selections, activates the previously created Selection submitting it to the AXEPTool Active Publication Active Rules/Selections, that is an instance of the AXCP Engine and Scheduler. The Actor could also specifies: <ul style="list-style-type: none"> <li>○ an activation period, which determines since when and for how long the objects of the Selection will be available on AXEPTool network</li> <li>○ etc...</li> </ul> </li> <li>2 Publication Tool Engine of AXEPTool, that is an instance of the AXCP Engine, elaborates the active Selections contained in the AXEPTool Active Publication Active Rules/Selections according to activation periods: <ol style="list-style-type: none"> <li>2.1 the Engine expands the Selection</li> <li>2.2 for each object which belongs to the expanded Selection <ol style="list-style-type: none"> <li>2.2.1 Engine publishes the object on the AXEPTool OUT AXDB</li> </ol> </li> </ol> </li> <li>3 Every time an object is published on the AXEPTool OUT AXDB, it advises the Publishing and Monitoring Objects which will broadcast the event to all its counterparts on the network</li> <li>4 If someone in the network is interested in one of the published object, then him/her AXEPTool P2P Active Selection Engine will download the object. The download process will be monitored by the AXEPTool Monitor</li> <li>5 If a yet-published (and still-published) AXMEDIS object is updated in the local AXDB. The local AXDB advises the Publication Tool Engine of AXEPTool that the object has been updated.</li> <li>6 Publication Tool Engine of AXEPTool controls in the rule script if the object belongs to one of the active Selections. If it does: <ol style="list-style-type: none"> <li>6.1 Publication Tool Engine of AXEPTool update the object contained into the AXEPTool OUT AXDB with the new version of it</li> <li>6.2 As described in Step 3 all the other peers in the network are advised</li> <li>6.3 If someone of them has downloaded the previous version of the object and is still interested in it, him/her AXEPTool P2P Active Selection Engine will automatically downloads the new version as described in Step 8 of the use case “Automatic load (and update) of AXMEDIS objects into local AXMEDIS Database from AXEPTool”</li> <li>6.4 The download will be monitored by the local AXEPTool Monitor</li> </ol> </li> <li>7 At the Selection expire time (if exists), Publication Tool Engine of AXEPTool removes the objects belonging to the Selection from the AXEPTool OUT AXDB.</li> <li>8 Every time an object is removed from the AXEPTool OUT AXDB (than no more available on the AXEPTool network), it advises the Publishing and Monitoring Objects. It will broadcast the event to all its counterparts on the network which shall remove the object from their AXEPTool IN AXDB</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Publication of AXMEDIS objects on AXEPTool can be also made using the AXMEDIS Workflow Manager
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.8 Automatic programme and publication of AXMEDIS objects on distribution channels

<b>UCId</b>	UC3.1.8
<b>Use case</b>	Automatic programme and publication of AXMEDIS objects on distribution channels
<b>Description</b>	An Actor wants to create a programme to be published for a community of End-Users.
<b>Actors</b>	Publisher, Distributor

<b>Assumptions</b>	The Actor has previously created a Selection which contains AXMEDIS objects contained within the local AXDB using the AXQS User Interface integrated within the Programme and Publication Rules Editor (see use case “Querying for AXMEDIS objects and Selection creation”). Otherwise the Selection can be one of those stored within the Repository of Publication Rules/Selections
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor creates a Publication Programme specifying: <ul style="list-style-type: none"> <li>o the Selection/AXOID of AXMEDIS objects to be published</li> <li>o the publication periods, i.e. periods of time in which the AXMEDIS objects will be available for the End-User</li> </ul> </li> <li>2 The Actor could save the Publication Rule into the Repository of Publication Programmes</li> <li>3 The Actor activates the Programme by submitting it either through Workflow or directly from the P&amp;P Editor to the P&amp;P Engine</li> <li>4 The Programme and Publication Engine related to the chosen distribution channel elaborates the active Rules: <ol style="list-style-type: none"> <li>4.1 Programme and Publication Engine expands the Selection contained in the Rule</li> <li>4.2 before publication time, Programme and Publication Engine, using Formatting Engine (AXCP), performs the needed adaptation on each object in the Selection. The adaptation request is made on the base of specified formatting rules known to the Programme and Publication Engine, which are strictly connected to distribution channel characteristics associated to the Engine. If the Publication Rule contains some formatting rules those overrides the defaults Engine will use known rules or request a rule</li> <li>4.3 at publication time, Programme and Publication Engine uploads all the adapted objects on the Distribution Server</li> <li>4.4 when the publication time expires, Programme and Publication Engine removes the objects uploaded at Step 4.3 from the Distribution Server (or, in some way, commands to the Distribution Server to remove them)</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Publication process can be activated using the AXMEDIS Workflow Manager
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.9 Acquisition of AXMEDIS objects from the distributor

<b>UCId</b>	UC3.1.9
<b>Use case</b>	Acquisition of AXMEDIS objects from the distributor
<b>Description</b>	An Actor wants to acquire one or more AXMEDIS objects choosing them from the Distributor’s program (see use case “Automatic programme and publication of AXMEDIS objects on distribution channels”)
<b>Actors</b>	End-User
<b>Assumptions</b>	The Actor is using an AXMEDIS compliant Client Viewer

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, using the Query Support for Client integrated within the Client Viewer, creates a query to consult the list of available AXMEDIS objects on Distribution Channels</li> <li>2 The Actor submits the query</li> <li>3 The Query Support for Client transmits the query to the Query Support for Distributions Channel and gets back the response</li> <li>4 The Query Support for Client displays the response to the Actor. In particular it shows the following information: <ul style="list-style-type: none"> <li>○ name, e.g. song title, movie title</li> <li>○ description, e.g. movie story/review</li> <li>○ usability and related price, e.g. price for each kind of available action on the object</li> <li>○ availability periods</li> <li>○ relevant metadata</li> <li>○ etc.</li> </ul> </li> <li>5 Before entirely downloading an object, the Actor could preview it if the object itself has been designed to allow this action</li> <li>6 Once the Actor has previewed the objects, he/she chooses the objects he/she wants to acquire, e.g. picking the related checkboxes (similar to Selection creation, see use case “Querying for AXMEDIS objects and Selection creation”)</li> <li>7 The Actor downloads the objects on his/her terminal</li> <li>8 The Actor pays the required fees and receives the licenses to use (accordingly to the acquired rights) the objects</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The objects are stored on some storage devices directly reachable by the Actor</li> <li>• The related licenses are stored in the Protection Manager Support (server side) to which the Client Viewer refers</li> <li>• The Actor can use at any time and everywhere the object accordingly to the acquired rights</li> </ul>
<b>Variations</b>	A lot of variations can be proposed on Step 7 and 8. These two steps are strongly dependant by the business model used by the distributor and its partners. The possible variations regards also the transmission model (streaming, downloading) and a lot of other aspects
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 3.1.10 Viewing/Using of AXMEDIS objects

<b>UCId</b>	UC3.1.10
<b>Use case</b>	Viewing/Using of AXMEDIS objects
<b>Description</b>	An Actor wants to view/use a previously-acquired AXMEDIS object on his/her terminal
<b>Actors</b>	End-User
<b>Assumptions</b>	The Actor has acquired one or more AXMEDIS objects from a Distributor (see use case “Acquisition of AXMEDIS objects”)
<b>Steps</b>	The Actor, using the Client Viewer, can perform all authorized action (i.e. acquired rights) on the object
<b>Post-conditions</b>	None
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	
<b>Issues</b>	None

## 4 AXMEDIS Object Editing

- AXMEDIS EDITORS, AS AUTHORING TOOLS
- AXMEDIS INTERNAL VIEWERS
- AXMEDIS VISUAL AND BEHAVIOURAL EDITOR
- AXMEDIS OBJECT EDITOR AND VIEWERS
- AXMEDIS TOOLS FOR USING/PRODUCING AXMEDIS OBJECTS IN OTHER CONTENT TOOLS

### 4.1 AXMEDIS Editors, as authoring tools

- Creation of a new AXMEDIS object
- Load and save AXMEDIS objects
- Navigating through AXMEDIS objects
- Adding AXMEDIS elements to an existing AXMEDIS object
- Extracting AXMEDIS elements
- Removing an element from an AXMEDIS Object
- Moving an element within the AXMEDIS Object
- Adding a resource
- Managing/Modifying a resources
- Navigating and understanding DRM rules and PAR

#### 4.1.1 Creation of a new AXMEDIS object

UCId	UC4.1.1
Use case	Creation of a new AXMEDIS object
Description	An actor wants to create a new AXMEDIS object from scratch.
Actors	Integrator, Designer
Assumptions	AXMEDIS Editor is opened
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when an actor clicks on the “New object” buttons within the AXMEDIS Editor main window</li> <li>2 The system creates a new software representation of an empty AXMEDIS object, i.e. an object which contains only the root element and an empty Item element.</li> <li>3 The system shows to the actor a hierarchical view of the object</li> <li>4 The actor can add digital resources and can apply to them content processing algorithms such as extractor of metadata, etc.</li> </ol>
Post-conditions	None
Variations	<ul style="list-style-type: none"> <li>• The actor could also click on “New...” within the “File” menu of the application</li> <li>• If the actor want to associate an AXMEDIS Object ID with the newly created object, the system has to request it to the AXMEDIS Object ID Generator through the Protection Manager Support</li> </ul>
Asynchronous actions	None
Design suggestions	None
Issues	It is suggested that fingerprint should be calculated only when an AXOID is associated to the object, i.e. when an object really becomes an AXMEDIS Object. After that, fingerprint and descriptors shall be calculated and communicated to the AXCS every time the AXMEDIS object is significantly changed.

#### 4.1.2 Load and save AXMEDIS objects

UCId	UC4.1.2
Use case	Load and save AXMEDIS objects

Description	An actor wants to load an AXMEDIS object and, after he/she has worked on it, he/she wants to save it
Actors	Aggregator
Assumptions	The Actor is working with the AXMEDIS Editor
Steps	<ol style="list-style-type: none"> <li>1 The Actor clicks on the “Open object” buttons within the AXMEDIS Editor main window</li> <li>2 The AXMEDIS Editor shows a dialog to allow the actor to choose which object he/she wants to open. The dialog contains a tabbed pane. The first pane allows the Actor to open an object stored on the file-system. The second integrates the AXQS to allow the Actor to search and open objects stored in the local AXDB</li> <li>3 The AXMEDIS Editor opens an object using the AXOM</li> <li>4 If errors did not occur at previous step, <ol style="list-style-type: none"> <li>4.1 The AXMEDIS Editor creates an Hierarchy Editor and Viewer showing the object</li> </ol> </li> <li>5 Else <ol style="list-style-type: none"> <li>5.1 The AXMEDIS Editor shows a dialog to inform the Actor about what did not work correctly</li> </ol> </li> <li>6 The Actor works with the object (see others use cases in this section)</li> <li>7 The Actor clicks on the “Save object” buttons within the AXMEDIS Editor main window or on “Upload on DB” buttons.</li> <li>8 The AXMEDIS Editor requests to the AXOM to save the object</li> <li>9 The AXOM validates the software representation of the object</li> <li>10 If the object is valid, and the Actor has the rights to save a new version of the object, the AXOM overwrites the old object with the new one (in case of file-system save) or performing the needed protocol in order to upload the new object on the DB.</li> </ol>
Post-conditions	None
Variations	<ul style="list-style-type: none"> <li>• The actor could also click on “Open...” within the “File” menu of the application</li> <li>• Instead of saving the object at step 7, the Actor want to save the object as, i.e. he/she wants to save the (modified) object in a new path. To do that step 7 should be replaced with: <ol style="list-style-type: none"> <li>7a. The Actor clicks on the “Save object as...” buttons within the AXMEDIS Editor main window</li> </ol> Step 10 does not change except that instead of overwriting the old version the (modified) object is saved in the chosen path </li> </ul>
Asynchronous actions	None
Design suggestions	None
Issues	None

#### 4.1.3 Navigating through AXMEDIS objects

UCId	UC4.1.3
Use case	Navigating through AXMEDIS objects
Description	An actor wants to navigate through the elements which compose the AXMEDIS object currently opened.
Actors	Integrator, distributor
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is opened</li> <li>• An object is opened within the AXMEDIS Editor</li> </ul>

Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor open an Hierarchy View of an opened object</li> <li>2 The Hierarchy View shows a explorer-like tree view</li> <li>3 The Actor could expand each tree node to view its children (if it contains someone)</li> <li>4 The Hierarchy View should show a specific context menu for each showed elements. For each element, the context menu should allow the Actor (if possible): <ol style="list-style-type: none"> <li>4.1 To manipulate the element/object structure (e.g. add a child/brother node, remove the element, etc...)</li> <li>4.2 To open alternative element views (e.g. DRM Editor and View, Behaviour Editor and View, Metadata Editor and View, etc...)</li> <li>4.3 Etc...</li> </ol> </li> </ol>
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

#### 4.1.4 Adding AXMEDIS elements to an existing AXMEDIS object

UCId	UC4.1.4
Use case	Adding AXMEDIS elements to an existing AXMEDIS object
Description	An Actor wants to add an element to the AXMEDIS object structure
Actors	Integrator, producer
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An Hierarchy View of the object is open</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor click with the right mouse button on an existing element</li> <li>2 The Hierarchy View shows the proper context menu to the actor</li> <li>3 The Actor chooses “Add element...” and then chooses the type of element he/she wants to add</li> <li>4 If necessary, the Hierarchy View shows to the Actor a dialog to fill the element attributes and options</li> <li>5 If the Actor confirms, the Hierarchy View requests to the AXOM to add the new element</li> <li>6 If the Actor has the needed rights: <ol style="list-style-type: none"> <li>6.1 the AXOM adds the element to the AXMEDIS object</li> <li>6.2 the Hierarchy View updates the displayed tree with the new added node</li> </ol> </li> <li>7 Else: <ol style="list-style-type: none"> <li>7.1 the AXOM informs the Hierarchy View about why the Actor can not add the element</li> <li>7.2 Hierarchy View shows a dialog to the Actor with the received error</li> </ol> </li> </ol>
Post-conditions	None
Variations	<ul style="list-style-type: none"> <li>• The Actor could click on “Add element...” within the “Edit” menu of the application instead of using the context menu</li> <li>• The Actor could also add an element as “brother” of an existing element instead as child of a given element. That should be possible by choosing “Insert after...”/“Insert before...” from the “Edit” menu or the context menu (of the reference element)</li> </ul>
Asynchronous actions	None
Design suggestions	None

Issues	<ul style="list-style-type: none"> <li>• Adds or insertion should be done without making validity controls because step-by-step consistency maintenance could bring usability lacks</li> <li>• By such way, a validity control is necessary when the actor wants to save the object</li> </ul>
--------	--

#### 4.1.5 Extracting AXMEDIS elements

UCId	UC4.1.5
Use case	Extracting AXMEDIS elements
Description	An Actor wants to create a new AXMEDIS Object from an element of the currently opened object
Actors	Integrator, producer, distributor
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor click with the right mouse button on an existing element</li> <li>2 The Hierarchy View shows the proper context menu to the Actor</li> <li>3 The Actor chooses “Extract element...”</li> <li>4 The Hierarchy View shows a dialog to allow the Actor to choose the new location (into the local file-system, into the AXMEDIS Database, etc...) where AXOM should store the extracted element</li> <li>5 If the Actor confirms: <ol style="list-style-type: none"> <li>5.1 the Hierarchy View requests to the AXOM to extract the element in the given location</li> <li>5.2 the AXOM, after it has checked the Actor’s rights, creates a new AXMEDIS Object in the location specified by the Actor maintaining the protection imposed by the DRM rules of the original object</li> </ol> </li> <li>6 The object will contains the selected element with all its relevant information, such as: DRM rules, metadata, etc...</li> </ol>
Post-conditions	The new object must works in respect of all the DRM rules associated with the original element
Variations	<ul style="list-style-type: none"> <li>• The Actor could click on “Extract element...” within the “Edit” (or “File”?) menu of the application instead of using the context menu</li> <li>• Extraction could be invoked when the system is opening an External or ActiveX Editor/Viewer. In such a way the system could pass an AXMEDIS object component, with all its related information, to another application maintaining protection of IP</li> </ul>
Asynchronous actions	None
Design suggestions	None
Issues	When the Actor wants to export an element, the AXOM has to consider all related information, both contained within the element or externally associated, e.g. through an MPEG21 Annotation element

#### 4.1.6 Removing an element from an AXMEDIS Object

UCId	UC4.1.6
Use case	Removing an element from an AXMEDIS Object
Description	An Actor wants to remove an element, and its sub-tree, from the object
Actors	Integrator, producer
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An Hierarchy View of the object is open and the object contains at least one element (other than root element)</li> </ul>

Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor click with the right mouse button on an existing element</li> <li>2 The Hierarchy View shows the proper context menu to the Actor</li> <li>3 The Actor choose “Remove”</li> <li>4 The Hierarchy View shows to the Actor a dialog to confirm the deletion</li> <li>5 If the Actor confirms <ol style="list-style-type: none"> <li>5.1 the Hierarchy View requests to AXOM to remove the element</li> <li>5.2 the AXOM, after it has checked the Actor’s rights, removes the element from the AXMEDIS Object and informs all the related views/editors about the change</li> <li>5.3 the informed views/editors updates the showed information</li> </ol> </li> </ol>
Post-conditions	None
Variations	The Actor could click on “Remove” within the “Edit” menu of the application instead of using the context menu
Asynchronous actions	None
Design suggestions	None
Issues	<ul style="list-style-type: none"> <li>• Deletion should be done without making validity controls because step-by-step consistency maintenance could bring usability lacks</li> <li>• By such way, a validity control is necessary when the actor wants to save the object</li> <li>• The operations have to be allowed by the license if the AXMEDIS object is protected</li> </ul>

#### 4.1.7 Moving an element within the AXMEDIS Object

UCId	UC4.1.7
Use case	Moving an element within the AXMEDIS Object
Description	An Actor clicks on an element, drags it on the desired new location and drops it. In that way, he/she moves the element in a new location within the object
Actors	Integrator, producer
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An Hierarchy View of the object is open</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor clicks on an element and drags it</li> <li>2 When the Actor drops the element, releasing the mouse button, the Hierarchy View controls if the chosen position is an allowed one.</li> <li>3 If the position is a valid one: <ol style="list-style-type: none"> <li>3.1 the Hierarchy View request to the AXOM to move the element in the new position</li> <li>3.2 the AXOM controls if the Actor is allowed to do the operation. In that case, the AXOM moves the element</li> </ol> </li> <li>4 Else, the Hierarchy View cancels the operation warning the Actor</li> </ol>
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	The position of an element in hierarchy of an AXMEDIS Object determines the semantic of the element with respect to the other elements in the object.

#### 4.1.8 Adding a resource

UCId	UC4.1.8
Use case	Adding a resource
Description	When an Actor chooses to add/insert, into an object, a Resource element, the Actor should choose which resource he/she wants to link to the object and in which way (by reference, including it within the object, etc...)
Actors	Integrator, producer

Assumptions	The Actor is adding a Resource element to the object (view use case “Adding AXMEDIS element”)
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor chooses as type of element to be inserted “Resource”</li> <li>2 The Hierarchy View shows a dialog to allow the Actor to choose which resource to add and the way to add it to the object, e.g. either by reference or by directly codifying it within the object</li> <li>3 The Hierarchy View requests to the AXOM to add the given resource</li> <li>4 The AXOM controls if the Actor has the needed rights on the AXMEDIS objects and on the resource. That is, if the resource is an AXMEDIS Object the AXOM will control if the Actor has the embedding rights on the resource</li> <li>5 If the control succeeds: <ol style="list-style-type: none"> <li>5.1 If the Actor chooses to include the resource <ol style="list-style-type: none"> <li>5.1.1 The AXOM codifies the resource in base64</li> <li>5.1.2 The AXOM includes the codified resource within the object</li> </ol> </li> <li>5.2 Else, the AXOM sets the “reference” attribute of the Resource element with value the location of the element</li> </ol> </li> </ol>
Post-conditions	None
Variations	A resource can be also a license, in that case the object becomes a governed object. In that case the License has to be verified consistent with the included PAR into the AXInfo.
Asynchronous actions	None
Design suggestions	None
Issues	None

#### 4.1.9 Managing/Modifying a resources

UCId	UC4.1.9
Use case	Managing/Modifying a resources
Description	An Actor wants to manage or modify a resource within the AXMEDIS object
Actors	Integrator, producer, etc.
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An Hierarchy View of the object is open</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor double-clicks on a Resource element (or an Item element which contains one resource, or a set of resources of the same type)</li> <li>2 The Hierarchy View asks to the AXOM to look for an editor/viewer associated with resource mime type attribute</li> <li>3 If there is no such editor/viewer, the Hierarchy View displays an error dialog</li> <li>4 Else <ol style="list-style-type: none"> <li>4.1 The AXOM locks the Resource element</li> <li>4.2 The AXOM certifies the editor/viewer</li> <li>4.3 The AXOM runs the editor/viewer as an independent thread</li> <li>4.4 The AXOM passes, through a secure channel, the extracted resource (view use case “Extract AXMEDIS elements”) to the editor/viewer</li> <li>4.5 After the editor/viewer has been closed, the AXOM unlocks the Resource element and update the previously extracted resource with the output of the editor/viewer</li> </ol> </li> </ol>
Post-conditions	None

Variations	<ul style="list-style-type: none"> <li>• The Actor could select a Resource element and then clicks on “Open resource...” within the “Edit” menu of the application instead of double-clicking on the element</li> <li>• When the AXOM is looking for an editor/viewer associated to a given MIME type, the AXOM has to search among three different kind of interfacing models which should be equivalent by actor point of view (except perhaps for the implemented functionalities): <ul style="list-style-type: none"> <li>○ Internal AXMEDIS Resource Editors/Viewers</li> <li>○ External Editor/Viewer AXMEDIS Plug-ins</li> <li>○ ActiveX Editor/Viewer AXMEDIS Plug-ins</li> </ul> </li> </ul>
Asynchronous actions	Since AXMEDIS Editor and resource plug-ins work as asynchronous processes, during plug-in execution, the actor could still work on the AXMEDIS object except for what concern the involved Resource element and its genealogy which have been locked before plug-in execution.
Design suggestions	None
Issues	A certified editor/viewer has to contain an Protection Processor and Protection Manager Client modules to insure DRM rules respect in all its parts. The editor/viewer has to control Actor’s rights (through a dedicated interface which is provided by the AXOM) on every Actor’s actions. Operations that lead to not allowed by licensing and operation leading to an invalid object due to conflicting licenses have to be verified and solved.

#### 4.1.10 Navigating and understanding DRM rules and PAR

UCId	UC4.1.10
Use case	Navigating and understanding DRM rules and PAR
Description	An Actor wants a visualization of the DRM rules and PAR (Potential Available Right), which allow the Actor to edit them, associated with an AXMEDIS objects or a component thereof
Actors	Integrator, producer, distributor
Assumptions	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An Hierarchy View of the object is open</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1 The use case begins when the Actor clicks with the right mouse button on an element</li> <li>2 The Hierarchy View shows the proper context menu to the Actor</li> <li>3 If enabled, the Actor chooses “DRM View...”</li> <li>4 The AXMEDIS Editor shows to the Actor a view which nicely represents the DRM rules and PAR associated with the selected element. Such view will allow the Actor to: <ul style="list-style-type: none"> <li>○ understand which rights he/she can exercise on the object</li> <li>○ understand which rights he/she can gain</li> <li>○ manipulate (add, remove, change, etc...), both in graphical and textual manner, DRM rules and PAR. Obviously, such manipulations can be made only by the object owner. Moreover, the Actor can not make changes which are in contrast with respect to DRM rules and PAR related to parts of his/her object which belong to another owner</li> <li>○ look for predefined rules set or template</li> <li>○ etc...</li> </ul> </li> </ol>
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

## 4.2 AXMEDIS Internal Viewers

- Invoking an internal viewer/editor
- Managing a digital resource by respecting the DRM in an Internal Viewer/Editor
- Closing an Internal viewer/editor

### 4.2.1 Invoking an internal viewer/editor

<b>UCId</b>	UC4.2.1
<b>Use case</b>	Invoking an internal viewer
<b>Description</b>	The actor wants to visualize a digital resource
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An hierarchical view of the object is open</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor clicks with the right mouse button on an digital resource</li> <li>2 The editor shows a proper context menu to the actor</li> <li>3 The actor chooses “View...”</li> <li>4 A proper viewer/editor is associated with the resource on the basis of MIME type</li> <li>5 The system sends an opening authorization request to the PMS (via AXOM)</li> <li>6 If PMS does not provide the authorization <ol style="list-style-type: none"> <li>6.1 The system displays an authorization failure message on screen</li> <li>6.2 The Use Case ends</li> </ol> </li> <li>7 The system performs the verification of the AXMEDIS Editor</li> <li>8 If the verification is not valid <ol style="list-style-type: none"> <li>8.1 The system displays a verification failure message on screen</li> <li>8.2 The Use Case ends</li> </ol> </li> <li>9 The system activates the proper internal viewer.</li> <li>10 The digital resource is sent to the viewer/editor (via The Protection Processor of AXOM)</li> <li>11 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Double click on the resource</li> <li>• System will automatically invoke internal viewers to visualize resources when the user wants to “play” the entire AXMEDIS object</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	End User usually can only “play” an AXMEDIS objects so only internal “viewers” should be invoked

### 4.2.2 Managing a digital resource by respecting the DRM in an Internal Viewer/Editor

<b>UCId</b>	UC4.2.2
<b>Use case</b>	Managing a digital resource by respecting the DRM in an Internal Viewer
<b>Description</b>	During the visualisation or a manipulation of a digital resource, the user could use some functionalities/commands (cut, paste, filtering, export, save, etc...) that could require the verification of DRM rule associated with the digital resource
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An internal viewer has been invoked by the system</li> </ul>

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to perform a command on the digital resource</li> <li>2 The system verifies the DRM of the resource (i.e. if the actor has the right to perform such command). This is done by means of the PMS client and its authorisation functionality.</li> <li>3 If the user is authorised             <ol style="list-style-type: none"> <li>3.1 The internal viewer/editor performs the command</li> </ol> </li> <li>4 Else             <ol style="list-style-type: none"> <li>4.1 The internal viewer/editor notifies a command failure message.</li> </ol> </li> <li>5 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The viewer continues to work</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The DRM verification is performed by the AXOM. by using PMS Client and PMS Server functionalities

#### 4.2.3 Closing an Internal viewer/editor

<b>UCId</b>	UC4.2.3
<b>Use case</b>	Closing an Internal Viewer
<b>Description</b>	After the visualisation of the digital resource, the user could decide to close the viewer
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An internal viewer has been invoked by the system</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to quit the internal viewer</li> <li>2 The user clicks with left mouse button on the close button of the system menu</li> <li>3 If the digital resource is changed             <ol style="list-style-type: none"> <li>3.1 The viewer/editor displays a dialog asking for the modification acceptance.</li> <li>3.2 If the actor does not discard the modification                 <ol style="list-style-type: none"> <li>3.2.1 The Viewer/Editor returns the modified resource</li> </ol> </li> <li>3.3 The viewer is closed</li> <li>3.4 The Use Case ends</li> </ol> </li> <li>4 The viewer is closed</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	If the resource is changed, the modified resource will substitute the original resource within the AXMEDIS object with the modified one.
<b>Variations</b>	The use could quit the viewer by selecting “Quit” in the menu bar
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 4.3 AXMEDIS Visual and Behavioural Editor

- Editing the visual scene for SMIL resource
- Editing the temporal information of media resources for SMIL resource
- Previewing the SMIL resources after editing
- Loading and saving the SMIL component into AXMEDIS object

The full description of the visual rendering and temporal behavior of an AXMEDIS object is comprised of a set of SMIL scenes related to each other. The sequence of SMIL scenes starts from the first SMIL scene (that is a resource in the AXMEDIS object as well) that is identified for its name: for example “Start”. It is not possible to have multiple SMIL files as “start SMIL scene” in an AXMEDIS object. The Start SMIL file is located in upper level of the hierarchical AXMEDIS structure/MPEG-21. The set of SMIL scenes related each other may be located in any place of the AXMEDIS object. Those contained in the components can be recalled and activated by those in the upper level of the hierarchy.

A SMIL scene is defined in terms of:

- Background: that can be realized as a visual element covering all the screen
- Elements (that initially can be rectangles). They may contain for rendering:
  - any audio visual element, that is video, audio, text, images, document, etc., that are contained into the AXMEDIS object; Please note that audible Elements do not have a visual rendering during the play, but may have visual rendering during the definition of the SMIL Scene;
  - buttons, which are represented by images, that can be clicked by the user to activate another SMIL scene. Therefore, they are a sort of hyperlinks to other SMIL Scene, thus each of them contain a reference to another SMIL scene in the AXMEDIS object.
- A link reference to the next SMIL scene to automatically pass to another SMIL scene when the Time Line of the Behavior is finished. The reference to the next SMIL scene can be also linked to the same SMIL scene thus creating a replication of the same scene forever. For example to present a video with two overlapped button waiting for the choice of the user.

In the end, any reference to SMIL scene may contains in alternative a reference to an AXMEDIS objects, that is an AXOID, in that case the Start SMIL file of the referred AXMEDIS object is executed.

Visual Elements of a SMIL scene can be overlapped each other and their temporal execution is defined by the behavior editor, that is a scheduler of their playing.

The Behavior Editor defines the time behavior and synchronization of elements in a single SMIL Scene. It has to allow defining in the time line for each element:

- start, stop, pause;
- periodic execution (it can be realized by calling again the same SMIL scene, or replicating the same resource several times along the time line or having multiple elements with associated with the same resource);
- hidden or visible/audible.

The definition of both Visual Elements and Behavior of SMIL scene implies the generation of the SMIL for that scene. In an AXMEDIS objects many of them may exist and can be related each other according to what has been defined above.

#### 4.3.1 Editing the visual scene for SMIL scene

##### 4.3.1.1 Creating and deleting an Element in a SMIL scene

<b>UCId</b>	UC4.3.1.1
<b>Use case</b>	creating and deleting a Element in a SMIL scene

<b>Description</b>	After viewing of the media resources, the user could decide to create the SMIL scene of a SMIL component based on the available media resources or to delete unwanted Element from the canvas.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>An internal visual editor and viewer has been invoked by the system</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to create/delete an Element in a SMIL scene component by choosing the options “create”/ “delete” of menu.</li> <li>The user left clicks on the palette on the left of the Visual Editor.</li> <li>When the Rectangular button is toggled, the user can create an Element (e.g., rectangular shape) on the canvas of the visual editor.</li> <li>The user left clicks in the canvas to create an element (e.g., rectangular shape) to represent the element area in the SMIL scene. Many of them can be arranged on the same scene and they can play the role of buttons or of windows to represent, images, video, audio, etc. So that some of them may not have a visual rendering but only an audible rendering.</li> <li>The user clicks “remove” to delete an Element from canvas.</li> </ol>
<b>Post-conditions</b>	<p>If the SMIL resource is changed, the modified resource will substitute the original resource within the AXMEDIS object with the modified one.</p> <p>If the Element in the SMIL scene is deleted, the resources associated to it will also disappear and the behaviour editor and viewer will change accordingly.</p> <p>On the other hand the resource in the AXMEDIS object will continue to exist.</p> <p>When a resource in the AXMEDIS object is removed from the AXMEDIS object, the all related SMIL Scene have to be verified to avoid the activation of missing resources.</p>
<b>Variations</b>	Double click on the existing SMIL resource; the system will automatically invoke visual editor and viewer to visualize the visual scene of the existing SMIL resources.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.1.2 Resizing and moving the Element in a SMIL scene

<b>UCId</b>	UC 4.3.1.2
<b>Use case</b>	Resizing and moving the Element in a SMIL scene
<b>Description</b>	After creating the Element (e.g., rectangular shape) for the visual scene, the user could decide to resize the Element
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>An internal visual editor and viewer has been invoked by the system</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to resize the Elements for the media resources.</li> <li>The user left clicks on the visual element (e.g., rectangular shape) on the canvas of the Visual Editor.</li> <li>When the Rectangular is selected, the user can resize the shape of the visual Element by dragging its edges in different directions to reach the desirable size.</li> <li>When the Rectangular is selected, the user can move the visual Element to any position on the canvas.</li> <li>the new version of the SMIL Scene is saved</li> </ol>
<b>Post-conditions</b>	The visual Element resulted to be changed in size and moved
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.1.3 Changing the background color of the visual Element

<b>UCId</b>	UC4.3.1.3
<b>Use case</b>	Changing the background color of the visual Element
<b>Description</b>	After creating the rectangular shape for the visual scene, the user could decide to change the background colour of the visual Element.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>An internal visual editor and viewer has been invoked by the system</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to change the background colour of visual elements.</li> <li>The user left clicks on the rectangular shape on the canvas of the Visual Editor.</li> <li>When the rectangular is selected, the user can choose the button or right click to select the option of changing the background colour of the visual element.</li> </ol>
Post-conditions	
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.1.4 Association of media resources with an Element of a SMIL Scene

<b>UCId</b>	UC4.3.1.4
<b>Use case</b>	Association of media resources with an Element of a SMIL Scene
<b>Description</b>	After creating the Element, the user could decide to associate the available media resources of the AXMEDIS object with the Elements of SMIL Scene.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>An internal visual editor and viewer has been invoked by the system</li> <li>The visual scene is already created using visual editor.</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to associate media resources with the Elements of SMIL scene</li> <li>The user left clicks on the rectangular to choose an Element for holding the media resources.</li> <li>The user can click on the right button of the mouse (or choose the menu on the frame, in both cases) with the options of “adding text”, “adding audio”, “adding video”, “adding image” to include different types of media resources.</li> <li>In alternative the user may browse the AXMEDIS hierarchy for selecting a resource and drag it to an Element and drop it for associating the resource to the Element in this manner.</li> <li>In alternative, the user can browse the media resources in the AXMEDIS object hierarchy and select one of them to associate it to the region.</li> </ol>
Post-conditions	<p>If the AXMEDIS resource is changed, the modified resource will substitute the original resource within the AXMEDIS object with the modified one.</p> <p>On the other hand the resource in the AXMEDIS object will continue to exist. When a resource in the AXMEDIS object is removed from the AXMEDIS object, the all related SMIL Scene have to be verified to avoid the activation of missing resources.</p>
<b>Variations</b>	Right click on the visual scene of SMIL resource; system will prompt a menu with the same options on the frame.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.2 Editing the temporal information of media resources

##### 4.3.2.1 Editing the unit and length of timeline

<b>UCId</b>	UC4.3.2.1
-------------	-----------

<b>Use case</b>	Editing the unit and length of timeline
<b>Description</b>	The user could edit the temporal information (time line) to display media components of the SMIL resource with behaviour editor and viewer; the user may decide the resolution and the time unit.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• The SMIL scene is already created and the AXMEDIS resources have already been associated with each Element using visual editor.</li> <li>• The behaviour editor and viewer is activated</li> </ul>
<b>Steps</b>	<p>1 The use case begins when the user wants to edit the time ruler by clicking the behaviour editor and viewer button.</p> <p>2. The user will have a contextual menu with a list of the properties of the timeline</p> <p>2 The user can edit the scale of time by inputting the time unit (in second, 10seconds, etc).</p> <p>3 The user can then edit the entire displaying time length of the timeline by changing the length of the ruler.</p> <p>4 After clicking “OK” of the contextual menu, the timeline will change according to the new information.</p> <p>5 The user can define which SMIL scene is to be activated after the execution of the current one. A list of SMIL scenes located to the AXMEDIS object under editing is provided and the user may select one of them. In the list of SMIL scene also the list of AXMEDIS objects is reported. When an AXMEDIS object is selected as successive SMIL scene, the Start SMIL file of the AXMEDIS object will be activated during the play.</p>
Post-conditions	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.2.2 Editing the displaying time boundary of each media resource

<b>UCId</b>	UC4.3.2.2
<b>Use case</b>	Editing the displaying time boundary of each media resource
<b>Description</b>	The user could edit the temporal boundary of each media resource
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• The SMIL scene is already created and the AXMEDIS resources have already been associated with each Element using visual editor.</li> <li>• The behaviour editor and viewer is activated</li> </ul>
<b>Steps</b>	<p>1 The use case begins when the user wants to edit the displaying time for the AXMEDIS resources associated to Elements in the SMIL scene by clicking the behaviour editor button.</p> <p>2 The behaviour editor and viewer will show the time line of the AXMEDIS resources and a temporal window of activation (a rectangular shape) along the time axis</p> <p>3 The user can modify the length and position of the rectangle on the time line to indicate the different starting time and display duration.</p> <p>4 For each instant of the rectangle on the time line it has to be possible to indicate if the AXMEDIS Resource has to be visualised or hidden;</p> <p>4. At the same time, it saves into the SMIL Scene into the AXMEDIS object.</p>
Post-conditions	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None

<b>Issues</b>	None
---------------	------

#### 4.3.3 Previewing the SMIL Scene after editing

<b>UCId</b>	UC4.3.3
<b>Use case</b>	Previewing the SMIL Scene after editing
<b>Description</b>	The user could preview the SMIL Scene after creating or editing using the visual and behaviour viewer.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>The SMIL scene is already created and the AXMEDIS resources have already been associated with each Element of the SMIL Scene or sequence of scenes by using visual editor.</li> <li>The temporal information of AXMEDIS resources for the elements of the SMIL Scene has been already edited using behaviour editor.</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to preview SMIL Scene or a sequence of them related to subsequence links via buttons of for direct connection after it is being created, edited or modified.</li> <li>The user could left click on the button “Preview” on the menu of the frame to preview the SMIL Scene by the internal SMIL player, starting from the current SMIL Scene under editing. And there would be a new frame popping up to show.</li> <li>The user could left click on the button “Stop” on the menu of the frame to stop previewing SMIL Scene.</li> <li>The user could close the frame to stop previewing SMIL Scene</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.3.4 Loading and saving the SMIL component into AXMEDIS object

<b>UCId</b>	UC4.3.4
<b>Use case</b>	loading and saving the SMIL component into AXMEDIS object
<b>Description</b>	The user could decide to load the SMIL component from AXMEDIS object to edit or save the SMIL component into AXMEDIS object after editing. These are the default values.
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>The visual editor is already activated and new SMIL component has been created/edited.</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to load the existent SMIL scene to edit or save it after editing by clicking load/save on the menu of frame.</li> <li>After saving, the SMIL component would enclosed inside a &lt;Component&gt; tag pair in AXMEDIS object.</li> <li>The save of new SMIL Scene has to be performed automatically providing the right position into the AXMEDIS object. The first SMIL scene of an AXMEDIS object saved is called Start (it is the starting point). The name can be changed later, but no more than one Start SMIL Scene may exist in each AXMEDIS component in the Root.</li> <li>The user can load a SMIL scene and display it by double clicking on it from the AXMEDIS hierarchy of the AXMEDIS Editor has to lead at the opening of the Visual Editor on that Scene.</li> <li>The simple double click on a SMIL Scene file into the AXMEDIS hierarchy of the AXMEDIS Player has to lead at the playing of the SMIL Scene.</li> </ol>
<b>Post-conditions</b>	

<b>Variations</b>	It has to be possible to save the SMIL Scene in a file into the hard disk of the computer. It has to be possible to load a SMIL Scene from a File located into the hard disk of the computer. It has to be possible to make operation of copy, cut and past of SMIL Scene file located into the AXMEDIS Hierarchy, from two or more different AXMEDIS objects/hierarchies opened at the same time by the AXMEDIS Editor. The same activity should be possible via drag anddrop (probably already possible)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.4 Navigation and hyperlinking with multiple SMIL Scenes

<b>UCId</b>	UC4.4
<b>Use case</b>	Navigation and hyperlinking with multiple SMIL Scenes
<b>Description</b>	The user during play will have the possibility of choosing or navigating among the available SMIL scenes with related elements and resources in the AXMEDIS object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	The set of SMIL scene is in the AXMEDIS object. An SMIL Scene named as Start is located into the root of the object. The SMIL Scenes are connected one to another, some buttons and/or menus have been created to establish relationships among the difference SMIL Scenes into the AXMEDIS object.
<b>Steps</b>	<p>1. The use case begins when:</p> <ul style="list-style-type: none"> <li>-- one SMIL scene is under execution and it includes some buttons (realised as Elements, to create from simple to more complex menus) at which another SMIL Scene is connected via a link and thus the user wants to pass to another SMIL Scene by clicking “link” on the Element which is at the bottom of the SMIL Scene. On the other hand, the user does not need to do anything if he does not want to link to other resources. In this case, the SMIL scene will arrive at concluding its Time Line and thus passing at executing the next SMIL Scene in any case. Please note that the next SMIL scene can be the same in execution, in that cases the reload should be avoided and the user may wait forever before taking a decision of clicking on the button.</li> <li>-- the SMIL Scene under execution complete the time line and pass to a next SMIL Scene as defined in Reference Link stated during the SMIL Scene editing.</li> </ul> <p>2. When “link” is activated (in any case), the next SMIL Scene with related Elements and resources is loaded to be played replacing the previous one with all its related Elements.</p>
<b>Post-conditions</b>	
<b>Variations</b>	The SMIL Scene in execution is changed passing to the next one.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.5 AXMEDIS Object Editor and viewers

##### 4.5.1 Opening annotations and comments of the media object

<b>UCId</b>	UC4.5.1
<b>Use case</b>	Opening annotations and comments of the media object

<b>Description</b>	The user could add some audio annotations or comments for the media object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>The object editor and viewer is activated.</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user selects a media object and clicks the object editor and viewer.</li> <li>A list of annotations and comments of this media object will be shown. Each of them will be related to the original resource with a specific link.</li> </ol>
Post-conditions	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

## 4.5.2 Adding annotations and comments of the media object

### 4.5.2.1 Adding audio annotations and comments of the media object

<b>UCId</b>	UC4.5.2.1
<b>Use case</b>	Adding audio annotations and comments of the media object
<b>Description</b>	The user could add some audio annotations or comments for the media object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	The object editor and viewer is activated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to add some comments or annotations for the media object. The user left clicks to select a media object in the AXMEDIS object resources</li> <li>The user clicks “adding audio comments”</li> <li>The user could record his/her own comments by using some available audio recorder/player</li> <li>The user clicks “OK” to save this operation</li> </ol>
Post-conditions	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

### 4.5.2.2 Adding text annotations and comments of the media object

<b>UCId</b>	UC4.5.2.2
<b>Use case</b>	Adding text annotations and comments of the media object
<b>Description</b>	The user could add some text annotations or comments for the media object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	The object editor and viewer is activated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>The use case begins when the user wants to add some comments or annotations for the media object. The user left clicks to select a media object in the AXMEDIS object resources</li> <li>The user clicks “adding text comments”</li> <li>The user could have a notebook to write his/her own comments.</li> <li>The user clicks “OK” to save this operation</li> </ol>
Post-conditions	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.

<b>Issues</b>	None.
---------------	-------

#### 4.5.2.3 Adding graphical annotations and comments of the media object

<b>UCId</b>	UC4.5.2.3
<b>Use case</b>	Adding graphical annotations and comments of the media object
<b>Description</b>	The user could add some graphical annotations or comments for the media object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	The object editor and viewer is activated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to add some comments or annotations for the media object. The user left clicks to select a media object in the AXMEDIS object resources</li> <li>2 The user clicks “adding graphical comments”</li> <li>3 The user could attach some pictures to this media object.</li> <li>4 The user clicks “OK” to save this operation</li> </ol>
<b>Post-conditions</b>	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

#### 4.5.3 Saving annotations and comments of the media object

<b>UCId</b>	UC4.5.3
<b>Use case</b>	Saving annotations and comments of the media object
<b>Description</b>	The user wants to save annotations or comments of the media object after editing
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• The object editor and viewer is activated.</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user has already changed the annotations and comments of the media object by clicking “saving”.</li> <li>2 The user will choose the directory and the name of the file to record the information of the annotations and comments of this media object</li> </ol> <p>The comments and annotations are saved on the disk and not into the AXMEDIS objects if now decided by the user and if the user has the right to do it.</p>
<b>Post-conditions</b>	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

#### 4.5.4 Removing annotations and comments of the media object

<b>UCId</b>	UC4.5.4
<b>Use case</b>	Removing annotations and comments of the media object
<b>Description</b>	The user could remove some annotations or comments for the media object
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• The object editor and viewer is activated.</li> <li>• Media object has already had the annotations and comments</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to delete the current annotations and comments of media object by clicking “removing”</li> <li>2 The annotations and comments will disappear and the file which record these annotations and comments will be deleted from the disk.</li> </ol>
<b>Post-conditions</b>	None.
<b>Variations</b>	None.

<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

#### 4.6 AXMEDIS tools for using/producing AXMEDIS Objects in other Content tools

- Invoking an external tool with a digital resource belonging to the AXMEDIS object
- Managing the digital resource by respecting the DRM in an external tool
- Closing an External Tool
- Updating a digital resource modified by an external tool
- Transferring a digital resource to an external tool

##### 4.6.1 Invoking an external tool with a digital resource belonging to the AXMEDIS object

<b>UCId</b>	UC4.3.1
<b>Use case</b>	Invoking external tools with a digital resource belonging to the AXMEDIS object
<b>Description</b>	The actor wants to use an external tool for content manipulation: (i) playing or rendering the resource, (ii) manipulating the resource by means of more complex functions provided by the external tool
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• AXMEDIS Editor is open</li> <li>• An object is opened within the AXMEDIS Editor</li> <li>• An hierarchical view of the object is open</li> <li>• The external tool is installed with a certified AXMEDIS plugin</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor clicks with the right mouse button on a resource</li> <li>2 The system shows the proper context menu to the actor</li> <li>3 The actor chooses “Open with...”</li> <li>4 The proper viewer/editor is associated with the resource on the basis of MIME type</li> <li>5 The AXMEDIS Editor performs the activation of the external tool if it is not already opened.</li> <li>6 After the activation a connection between AXMEDIS Editor and AXMEDIS plugin inside the external tool is established</li> <li>7 The AXMEDIS Editor perform the verification of the AXMEDIS Plug-in</li> <li>8 If the verification is not valid <ol style="list-style-type: none"> <li>8.1 The system displays an verification failure message on screen</li> <li>8.2 The Use Case ends</li> </ol> </li> <li>9 After the verification, a negotiation among DRM enforcement capabilities takes place</li> <li>10 Inside the editor a built-in logic can decide if the resource transfer is risky in terms of DRM since the external tool cannot enforce some specific action</li> <li>11 If the negotiation ends with a negative response <ol style="list-style-type: none"> <li>11.1 The system displays an negotiation failure message on screen</li> <li>11.2 The Use Case ends</li> </ol> </li> <li>12 The system transfer the digital resource to the plug-in (by using AXOM for saving an loading)</li> <li>13 The plug-in provides the digital resource to be represented in the external tool data model (see “Transferring a digital resource to an external tool” Use Case)</li> <li>14 The external tool shows the digital resource</li> <li>15 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None

<b>Variations</b>	<ul style="list-style-type: none"> <li>• Double click on the resource for automatic call of the tool by means MIME mechanism (default association)</li> <li>• A table with the available tools is associated with the resource, the system shows such table and the user selects the proper tool</li> <li>• System will automatically use external tools to visualize resources when the user wants to “play” the entire AXMEDIS object and, for example, no adequate internal viewers are available</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	End User usually can only render an AXMEDIS objects so only external “viewers” should be invoked

#### 4.6.2 Managing the digital resource by respecting the DRM in an external tool

<b>UCId</b>	UC4.3.2
<b>Use case</b>	Managing the digital resource by respecting the DRM in an external tool
<b>Description</b>	During the visualisation or a manipulation of a digital resource, the user could use functionalities (cut, paste, filtering, export, save, etc...) that could require the verification of some DRM rules
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An external tool has been invoked by the system</li> <li>• The external tool uses its AXMEDIS plug-in</li> <li>• The digital resource has been loaded in the AXMEDIS Data Model inside the plug-in and the resource is ready to be used.</li> <li>• The communication with the AXMEDIS Editor is active via plug-in</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to execute a command provided by the external tool</li> <li>2 The external tool communicates the request of command execution to the AXMEDIS plug-in</li> <li>3 The AXMEDIS plug-in verifies the DRM of the resource (i.e. if the actor has the right to perform such command). The license can impose limitations on the usage of the object into external editors. In addition, the operation requested can be not safe on that tool, the AXMEDIS plugin knows which are the safe operations (those that does not violate the license and that allow to trace the activities). In these cases the requested action cannot be granted.</li> <li>4 If the actor is authorised <ol style="list-style-type: none"> <li>4.1 The AXMEDIS plug-in authorises the External tool to perform the command</li> </ol> </li> <li>5 Else <ol style="list-style-type: none"> <li>5.1 The AXMEDIS plug-in does not authorise the external tool to execute the command and notifies a command failure message.</li> </ol> </li> <li>6 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The external tool is still working
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	This use case is only viable for those tools in which the DRM of the requested operation is controllable via the AXMEDIS plug-in. The profile of the AXMEDIS plug in has to declare which kind of content process and security level is present in the External Editor/tool. In general if the license do not accept the object to be processed by such an External Editor-tool the operation will result not feasible.

#### 4.6.3 Closing an External Tool

<b>UCId</b>	UC4.3.3
-------------	---------

<b>Use case</b>	Closing an external tool
<b>Description</b>	After the visualisation or manipulation phase of the digital resource by means of an external content editor or tool, the user could decide to close such tool
<b>Actors</b>	End User, Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An external tool has been invoked by the system</li> <li>• The external tool uses the AXMEDIS plug-in</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user wants to quit the external tool</li> <li>2 The user clicks with left mouse button on the close button of the external tool menu</li> <li>3 If the digital resource is changed <ol style="list-style-type: none"> <li>3.1 The tool displays a dialog asking for the modification acceptance.</li> <li>3.2 If the actor does not discard the modification <ol style="list-style-type: none"> <li>3.2.1 See the “Updating a digital resource modified by an external tool” Use Case</li> </ol> </li> <li>3.3 The tool is closed</li> <li>3.4 The Use Case ends</li> </ol> </li> <li>4 The tool is closed</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The modified digital resource has been updated in the AXMEDIS object</li> <li>• The external tool has been closed.</li> </ul>
<b>Variations</b>	The use could quit the tool by selecting “Quit” in the menu bar.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.6.4 Updating a digital resource modified by an external tool

<b>UCId</b>	UC4.3.4
<b>Use case</b>	Updating a digital resource modified by an external tool
<b>Description</b>	After that a digital resource has been modified by means of an external content editor or tool, the AXMEDIS object has to be updated with the new version of the digital resource
<b>Actors</b>	AXMEDIS Plug-in, system
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An external tool has been invoked by the system</li> <li>• The external tool uses the AXMEDIS plug-in</li> <li>• The digital resource has been modified</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an updating command is invoked (to update a modified digital resource command)</li> <li>2 The plug-in provides the updating of the digital resource in the AXMEDIS object transferring the new digital resource from the external tool data model to the AXMEDIS object</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The digital resource has been updated successfully inside the AXMEDIS object</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 4.6.5 Transferring a digital resource to an external tool

<b>UCId</b>	UC4.3.5
<b>Use case</b>	Transferring a digital resource to an external tool
<b>Description</b>	After that an external tool has been invoked, the digital resource to be modified has to be transferred from the AXMEDIS object to the external tool data model

<b>Actors</b>	AXMEDIS Plug-in, AXMEDIS Editor, System
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• An external tool has been invoked by the system</li> <li>• The external tool uses the AXMEDIS plug-in</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the user has initiated an external tool session of resource usage by choosing “Open with...”</li> <li>2 The AXMEDIS plug-in negotiate with the editor if the transfer of the resource does not put in risk the latter. The negotiation will be based on the information which are provided by the AXMEDIS plug-in about the DRM enforcement capability. The AXMEDIS plug-in (certified) is aware about which are the safe action (those that does not violate the license and that allow to trace the activities).</li> <li>3 If the negotiation success the digital resource can be transferred to the external tool</li> <li>4 The AXMEDIS editor packs the resource inside an temporary object (file) created on-the-fly. The temporary object is loaded by the plug-in and the contained resource is extracted And passed to the external tool resource model.</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The digital resource has been transferred successfully to the external tool data model</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The transferring of the digital resource can be performed from the AXMEDIS Editor to the AXMEDIS Plug-in in a safe manner, for instance by encrypting the digital resources. The resource can be passed via a temporary file. All content processing operations, such as resource conversion, or others, have to be allowed by a specific valid license if the object is protected.

## 5 AXMEDIS Plug in definition

### 5.1.1.1 Defining a AXCP plugin

<b>UCId</b>	UC5.1.1.1
<b>Use case</b>	Defining a AXCP plugin
<b>Description</b>	The UC describes main steps related to the definition of an AXCP pluing
<b>Actors</b>	A programmer
<b>Assumptions</b>	The actor has identified a set of content processing algorithms to be developed and works with the AXMEDIS Framework. The actor uses the xml schema related to edit the profile of functions that the plugin will expose
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor develops a library of functions according to the identified content processing algorithms</li> <li>2 The actor starts to define and implement a dynamic library for the development of AXMEDIS Plugin</li> <li>3 For each functions of the library <ol style="list-style-type: none"> <li>3.1 The Actor defines and develops an AXCP Function class. The class has to be called using the name of the native content processing function</li> <li>3.2 The Actor maps the set of in/out parameters of the native function signature into the AXCP Parameter classes according to data types</li> <li>3.3 The Actor implements a method called “execute” where he puts the call to the native content processing function of library</li> </ol> </li> <li>4 The Actor implements an entry function that manages a selector of single instances of AXCP Functions</li> <li>5 The Actor has to exports the entry function as entry point of a dynamic library</li> <li>6 The Actor has to edit an xml file describing each function of the plugin</li> <li>7 The Actor puts the plugin and the xml in the plugin directory of the AXCP tools</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• A <i>dll</i> or <i>so</i> library is created</li> <li>• An xml profile has been associated with the plugin</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 6 AXMEDIS Production Tools

### 6.1 Automatic Production Tools

#### 6.1.1 AXMEDIS Content Processing Engine

##### 6.1.1.1 Firing an AXCP rule

<b>UCId</b>	UC6.1.1.1
<b>Use case</b>	Firing an AXCP rule
<b>Description</b>	The UC describes steps related to the firing of a rule in the AXCP Rule Engine
<b>Actors</b>	Internal Scheduler of the AXCP Rule Engine
<b>Assumptions</b>	The Internal Scheduler of the AXCP Rule Engine has almost an activated AXCP rule
<b>Steps</b>	<ol style="list-style-type: none"> <li>8 The Internal Scheduler periodically checks if the firing condition of rules are verified.</li> <li>9 If the firing conditions are verified               <ol style="list-style-type: none"> <li>9.1 A Remote Executor is associated with the AXCP rule</li> <li>9.2 The rule is sent to the Executor to be run</li> </ol> </li> </ol>
<b>Post-conditions</b>	The fired rule is running on the Remote Executor
<b>Variations</b>	<p>The AXMEDIS Workflow manager sends a request of running a specific AXCP rule.</p> <p>The association of Remote Executor with the rule is performed by matching the required capabilities by the rule with the executor profile capabilities</p>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

##### 6.1.1.2 Searching for a rule Executor

<b>UCId</b>	UC6.1.1.2
<b>Use case</b>	Searching for a rule Executor
<b>Description</b>	The UC describes steps related to the discovering of an AXCP Rule Executor in the AXCP Rule Engine Grid Environment
<b>Actors</b>	Internal Scheduler of the AXCP Rule Engine
<b>Assumptions</b>	AXCP Rule Executors are running in the GRID Environment
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The internal scheduler performs periodically a network exploration in order to discover new AXCP Remote Executor</li> <li>2 If a new executor is found               <ol style="list-style-type: none"> <li>2.1 the Remote Executor is put into the list of available executor</li> <li>2.2 The rule is sent to the Executor to be run</li> </ol> </li> </ol>
<b>Post-conditions</b>	A new AXCP Remote Executor is available
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

##### 6.1.1.3 Automatic production

<b>UCId</b>	UC6.1.1.3
<b>Use case</b>	Automatic production
<b>Description</b>	Some AXMEDIS objects are created automatically
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	An active content processing rule is ready to be executed

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case begins when the Internal Scheduler of the AXCP Rule Engine activates a rule from the AXCP Rules List.</li> <li>2. The Internal Scheduler sends a Rule execution request and the corresponding rule to the selected AXCP Rule Executor.</li> <li>3. The AXCP Rule Executor executes the submitted rules by: <ol style="list-style-type: none"> <li>3.1. recovering all the specified AXMEDIS objects from AXMEDIS Database</li> <li>3.2. verifying the compatibility of DRM and licensing</li> <li>3.3. compounding AXMEDIS objects as described into selected rule</li> <li>3.4. interacting with Formatting, Fingerprint, Adaptation and Protection tools</li> <li>3.5. storing all new created AXMEDIS objects into AXMEDIS Database (AXMEDIS Objects repository)</li> </ol> </li> <li>4. If the AXCP Rule Executor runs correctly the rule <ol style="list-style-type: none"> <li>4.1. sends an End Process notification to the Internal Scheduler</li> </ol> </li> <li>5. otherwise it sends the occurred errors to the Internal Scheduler</li> <li>6. The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>o The AXCP Rule Engine receives a request coming from the AXMEDIS Workflow Manager.</li> <li>o The AXCP Rule Engine send a notification or the occurred error to the AXMEDIS Workflow Manager.</li> <li>o The formatting process can be executed by an external tool if specified in the formatting rule</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.4 Verification of the compatibility of DRM associated with digital resources

<b>UCId</b>	UC6.1.1.4
<b>Use case</b>	Verification of the compatibility DRM associated with digital resources
<b>Description</b>	An AXCP rule could include the verification request of DRM rules related to all digital resources with a DRM target specified by the Content Integrator. In this case The AXCP Engine verifies that DRM rules are compatible with the DRM rules and/or conditions specified in the rule
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The DRM rules of digital resources related to the Selection of objects are available
<b>Steps</b>	<ol style="list-style-type: none"> <li>3 The Use Case starts when the AXCP Rule Executor has to verify if the set of DRM rules match the DRM target specified in the rule.</li> <li>4 If DRM are not compatible with the DRM and/or conditions specified in the rule. <ol style="list-style-type: none"> <li>4.1 The execution fails and a failure notification is generated</li> <li>4.2 The Use Case ends.</li> </ol> </li> <li>5 The AXCP Rule Executor continues the rule execution</li> <li>6 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The current composition is interrupted.
<b>Variations</b>	The AXCP Rule Engine send a notification or the occurred error to the AXMEDIS Workflow Manager.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.1.1.5 Verification of rights for digital resources**

<b>UCId</b>	UC6.1.1.5
<b>Use case</b>	Verification of rights for digital resources
<b>Description</b>	An AXCP rule could include the verification request of rights related to all digital resources. In this case, the AXCP Engine verifies that rights are compatible with the rights target specified in the rule.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The DRM rules of digital resources related to the Selection of objects are available
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the AXCP Rule Executor has to verify if the set of rights match the rights specified in compositional rule.</li> <li>2 If rights are not compatible with the rights specified in the rule. <ol style="list-style-type: none"> <li>2.1 The execution fails and a failure notification is generated</li> <li>2.2 The Use Case ends.</li> </ol> </li> <li>3 The AXCP Rule Executor continues the rule execution</li> <li>4 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The current composition is interrupted.
<b>Variations</b>	The AXCP Rule Engine send a notification or the occurred error to the AXMEDIS Workflow Manager.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.1.1.6 Embedding a digital resource in the new AXMEDIS object**

<b>UCId</b>	UC6.1.1.6
<b>Use case</b>	Embedding a digital resource in the new AXMEDIS object
<b>Description</b>	AXCP Rule Engine embeds physically or by reference one or more digital resource in the new AXMEDIS object.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the AXCP Rule Executor has to embed a digital resource in the new AXMEDIS object</li> <li>2 If the embedding option is “physically” <ol style="list-style-type: none"> <li>2.1 The rule executor sends an embedding request and the resource to the AXOM</li> </ol> </li> <li>3 Else the composition engine sends an embedding request and the reference of resource to the AXOM</li> <li>4 The resource is embedded</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The resource or the reference is embedded in the AXMEDIS object
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.1.1.7 New AXMEDIS objects generation**

<b>UCId</b>	UC6.1.1.7
<b>Use case</b>	New AXMEDIS object generation
<b>Description</b>	AXCP Rule Engine creates one or more new AXMEDIS objects and assigns them a new Object ID
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	None

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the AXCP Rule Executor creates a new AXMEDIS object following an AXCP rule</li> <li>2 The AXCP Rule Executor asks for a new Object ID to the AXMEDIS OID Generator.</li> <li>3 The ID is applied to the new object.</li> <li>4 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The Object is created and a new ID has been assigned
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.8 Fingerprint estimation of a digital resource

<b>UCId</b>	UC6.1.1.8
<b>Use case</b>	Fingerprint estimation of a digital resource
<b>Description</b>	If a fingerprint request for a digital resource is specified in the AXCP rule the AXCP Rule Engine interacts with the Fingerprint tool asking for fingerprint estimation . The Fingerprint tool will return the content descriptors related to the digital resource.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The digital resource is available physically during the composition process
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when a fingerprint estimation request for the digital resource is specified in the AXCP rule.</li> <li>2 The digital resource is sent to the Fingerprint tool</li> <li>3 The Fingerprint tool returns the content descriptors associated with the digital resource.</li> <li>4 The content descriptors are inserted as metadata associated with the digital resource.</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.9 Formatting of AXMEDIS Objects

<b>UCId</b>	UC6.1.1.9
<b>Use case</b>	Formatting of AXMEDIS Objects.
<b>Description</b>	If an automatic formatting request for an AXMEDIS Object is specified in the AXCP rule, the AXCP Rule Engine interacts with the Format Tool. The Format Tool will return the format descriptors most suitable for the digital resources contained in the Object.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The AXMEDIS Object is available during the composition process; profiles for user preferences, device capabilities and delivery context should also be available.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case starts when an automatic format request for an AXMEDIS Object is specified in the AXCP rule.</li> <li>2. The AXMEDIS Object is sent to the Format Tool</li> <li>3. Profiles for user preferences, device capabilities and delivery context are sent to the Format Tool</li> <li>4. The AXCP Rule Executor asks the Format Tool for automatically selecting a template.</li> <li>5. The Format Tool returns an ordered list of descriptors for most suitable templates.</li> <li>6. The AXCP Rule Executor asks the Format Tool for automatically selecting a style-sheet for the first template of the list.</li> <li>7. The Format Tool returns an ordered list of descriptors for most suitable style-sheets.</li> <li>8. The AXCP Rule Executor asks the Format Tool for optimizing the first style-sheet of the list.</li> <li>9. The Format Tool returns the descriptor of the optimized style-sheet.</li> <li>10. The AXCP Rule Executor requests the creation of the resulting format.</li> <li>11. The resulting format descriptor is associated with the Object.</li> <li>12. The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.10 Adaptation of a digital resource

<b>UCId</b>	UC6.1.1.10
<b>Use case</b>	Adaptation of a digital resource
<b>Description</b>	If an adaptation request for a digital resource is specified in the AXCP rule the AXCP Rule Engine interacts with the Adaptation tool. The Adaptation tool will perform the adaptation specified in the composition rule for the a given digital resource.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The digital resource is available physically during the composition process
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when an adaptation request for the digital resource is specified in the AXCP rule</li> <li>2 The digital resource is sent to the adaptation tool</li> <li>3 The Adaptation tool returns the adapted resource</li> <li>4 The Use Case ends</li> </ol>
<b>Post-conditions</b>	The initial digital resource is adapted on the basis of adaptation request specified in the rule
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.11 Protection of the new AXMEDIS object

<b>UCId</b>	UC6.1.1.11
<b>Use case</b>	Protection of the new AXMEDIS object
<b>Description</b>	If a protection request for the new AXMEDIS object is specified in the AXCP rule the AXCP Rule Engine interacts with the Protection tool. The Protection tool will create an AXMEDIS protected object.

<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The digital resource is available physically during the composition process
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the rule executor has to perform a fingerprint estimation.</li> <li>2 The digital resource is sent to the Fingerprint tool</li> <li>3 The Fingerprint tool returns the content descriptors associated with the digital resource.</li> <li>4 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.12 Merging component's DRM/PAR rules into a new AXMEDIS object

<b>UCId</b>	UC6.1.1.12
<b>Use case</b>	Merging component's DRM/PAR rules into a new AXMEDIS object
<b>Description</b>	AXCP Rule Engine create a new AXMEDIS objects and merge component's DRM/PAR rules to create a new DRM/PAR rule
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the AXCP Rule Executor has to generate the DRM/PAR for the composite AXMEDIS object</li> <li>2 AXCP Rule Executor merges component's DRM/PAR rules into the new AXMEDIS Objects</li> <li>3 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>o The DRM/PAR of the whole new object depends on what could be the intersection of DRM/PAR rules related to each component</li> <li>o The actor can modify the DRM/PAR rule of the new AXMEDIS object without modify the DRM/PAR rules of the components, but they have to respect the existing . The verification can be invoked by exploiting services of the PMS client in conjunction with the AXOM</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.1.1.13 External Tools execute formatting operations

<b>UCId</b>	UC6.1.1.13
<b>Use case</b>	External Tools execute formatting operations
<b>Description</b>	If a request of services provided by formatting external tools is specified in the AXCP rule the AXCP Rule Engine will interact with the External Formatting tools. The External Tools will format or perform specific script languages. The external tool will be able to perform also adaptations specified in the formatting rule for digital resources.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment
<b>Assumptions</b>	The digital resources to be formatted are available physically during the formatting process

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the rule executor has to perform an external call to a formatting tool specified in the rule</li> <li>2 The Rule Executor sends the digital resources and parameters specified in the external call to the external tool</li> <li>3 The external tool performs functions specified in the rule</li> <li>4 The external tool returns formatted digital resources</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Parameters involved in the external call could be: user profile, channel distribution and device profile, file format, scripting code executable by the external tool, etc...
<b>Issues</b>	None

## 6.1.2 AXCP Rules Editor

### 6.1.2.1 Create a new AXCP rule

<b>UCId</b>	UC6.1.2.1
<b>Use case</b>	Create a new AXCP rule
<b>Description</b>	An Actor wants to create a new AXCP rule
<b>Actors</b>	Content owner, Content Integrator, Content Distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor creates a Selection of digital resources based on queries to the AXMEDIS Database</li> <li>2 The Actor defines the set of parameters necessary to run the rule</li> <li>3 The Actor defines the plugins/external tools to be used</li> <li>4 The Actor rules how these resources have to be processed</li> <li>5 The Actor stores the created rule into the Rule repository</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ol style="list-style-type: none"> <li>1) The Actor defines a Selection by writing in the rule the scripting code (AXCP Rule Language) for queries to be executed when the rule will be run</li> <li>2) The Actor can define a rule or writing it as scripting code (AXCP Rule Language) or in a Visual way.</li> </ol>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.1.2.2 Search and Select an AXCP rule

<b>UCId</b>	UC6.1.2.2
<b>Use case</b>	Search and Select an AXCP rule
<b>Description</b>	An Actor wants to select a specific compositional rule he should be enabled to make some search or browsing, they are organized in some ordering.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor
<b>Assumptions</b>	None

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor search into the Repository of AXCP Rules a specific AXCP rule</li> <li>2 The rules are ordered in some manner and simple queries can be performed</li> <li>3 If the Actor founds the rule can : <ol style="list-style-type: none"> <li>3.1 Use it to create a compounded AXMEDIS object</li> <li>3.2 Modify it <ol style="list-style-type: none"> <li>3.2.1 Then the Actor stores the new rule into the Repository by AXCP Rule Editor</li> <li>3.2.2 Use the new rule to create a compounded AXMEDIS object</li> </ol> </li> </ol> </li> <li>4 If the Actor doesn't found the rule can create a new one</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.1.2.3 Activating an AXCP rule

<b>UCId</b>	UC6.1.2.3
<b>Use case</b>	Activating an AXCP rule
<b>Description</b>	An Actor wants to activate an AXCP rule
<b>Actors</b>	Content owner, Content Integrator, Content Distributor
<b>Assumptions</b>	The AXCP Rule Editor can access to the Rules Repository
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor searches into the Repository of Rules a specific AXCP rule</li> <li>2 If the Actor doesn't found the rule <ol style="list-style-type: none"> <li>2.1 The Actor can create a new one</li> </ol> </li> <li>3 The Actor selects "Activate" function</li> <li>4 The rule is sent to the Active Rules Repository of the AXCP Rule Engine</li> </ol>
<b>Post-conditions</b>	The rule is added to the set of Active Rule
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.1.2.4 Debugging/simulation of an AXCP rule

<b>UCId</b>	UC6.1.2.4
<b>Use case</b>	Debugging/Simulation of an AXCP rule
<b>Description</b>	An Actor wants to debug or simulate an AXCP rule
<b>Actors</b>	Content owner, Content Integrator, Content Distributor
<b>Assumptions</b>	The script associated with the AXCP rule is available in the Rule Editor
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case starts when the Actor wants to debug the script</li> <li>2 The Rule Editor enters in the Debugging/Simulation Mode</li> <li>3 During the debugging mode the Actor can: <ol style="list-style-type: none"> <li>3.1 Check the statements of script step by step</li> <li>3.2 Control the values of current variables</li> <li>3.3 Add/Remove breakpoints</li> <li>3.4 Step Over lines and Trace Into functions</li> <li>3.5 Exit from the debugging mode</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	The AXCP rule is simulated during the debugging session in order to extract information regarding the complexity, CPU request, workload, etc....
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None

<b>Issues</b>	None
---------------	------

## 6.2 Formatting Tools

### 6.2.1 Automatic Formatting Tools

#### 6.2.1.1 Automatic formatting process

<b>UCId</b>	UC6.2.1.1
<b>Use case</b>	Automatic formatting process
<b>Description</b>	An Actor wants to format an AXMEDIS Object, using the automatic formatting facilities.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its digital resources are accessible to the Format Tool; profile descriptors for user preferences, device capabilities and delivery context should also be available.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case begins when the Actor sends a formatting request to the Format Tool.</li> <li>2. The Actor performs an automatic template selection.</li> <li>3. The Actor performs an automatic style-sheet selection and optimization.</li> <li>4. The Actor performs an automatic style-sheet optimization.</li> <li>5. The Actor performs a format creation.</li> <li>6. The Actor receives the resulting format descriptor.</li> <li>7. The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A format for the Object has been produced.
<b>Variations</b>	The Actor may stop the process at any step and go back to previous steps.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.2.1.2 Automatic template selection

<b>UCId</b>	UC6.2.1.2
<b>Use case</b>	Automatic template selection
<b>Description</b>	An Actor wants to select a ready-made template for an AXMEDIS Object, using the automatic formatting facilities.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its digital resources are accessible to the Format Tool; profile descriptors for user preferences, device capabilities and delivery context should also be available.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case begins when the Actor sends a template selection request to the Format Tool.</li> <li>2. The Actor provides resource references.</li> <li>3. The Actor may provide Profile references (User, Device and Context Profiles)</li> <li>4. The Actor may provide information about the type of document to be created.</li> <li>5. The Actor may provide information about the preferred output format.</li> <li>6. The Actor may provide information about the target device(s).</li> <li>7. The Actor may provide indications about the category of each resource.</li> <li>8. The Tool produces an ordered list of descriptors for best templates.</li> <li>9. The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A list of templates has been selected.
<b>Variations</b>	The Actor may bypass the selection logic providing a template ID: in this case the list will only contain the template with the given ID.
<b>Asynchronous actions</b>	None

<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.2.1.3 Automatic style-sheet selection

<b>UCId</b>	UC6.2.1.3
<b>Use case</b>	Automatic style-sheet selection
<b>Description</b>	An Actor wants to select a ready-made style-sheet, suitable for the given template, using the automatic formatting facilities.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its resources are accessible to the Format Tool; profile descriptors for user preferences, device capabilities and delivery context should also be available. A template has also been selected.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case begins when the Actor sends a style-sheet selection request to the Format Tool.</li> <li>2. The Actor provides resource references.</li> <li>3. The Actor may provide Profile references (User, Device and Context Profiles).</li> <li>4. The Actor may give information about the target device(s).</li> <li>5. The Tool produces an ordered list of descriptors for best style-sheets.</li> <li>6. The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A list of style-sheets has been selected.
<b>Variations</b>	The Actor may bypass the selection logic providing a style-sheet ID: in this case the list will only contain the style-sheet with the given ID.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.2.1.4 Automatic style-sheet optimization

<b>UCId</b>	UC6.2.1.4
<b>Use case</b>	Automatic style-sheet optimization
<b>Description</b>	An Actor wants to optimize a style-sheet, using the automatic formatting facilities.
<b>Actors</b>	A Rule Executor in the AXCP Rule Engine Grid Environment.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its resources are accessible to the Format Tool; profiles for user preferences, device capabilities and delivery context should also be accessible. A template and a style-sheet have also been selected.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor sends a style-sheet optimization request to the Format Tool.</li> <li>2 The Actor provides resource references.</li> <li>3 The Actor may provide Profile references (User, Device and Context Profiles).</li> <li>4 The Actor may give information about the target device(s).</li> <li>5 The Tool produces optimized values for parameters defined in the style-sheet and creates a descriptor for the optimized style-sheet.</li> <li>6 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	The style-sheet has been optimized.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.2.1.5 Format creation**

<b>UCId</b>	UC6.2.1.5
<b>Use case</b>	Format creation
<b>Description</b>	An Actor wants to use a template and an optimized style-sheet for creating a format descriptor.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor, a Rule Executor in the AXCP Rule Engine Grid Environment.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its resources are accessible to the Format Tool. A template and a style-sheet have also been selected and their descriptors are available to the Format Tool.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor sends a format creation request to the Format Tool.</li> <li>2 The Actor provides resource references.</li> <li>3 The Actor provides template and style-sheet descriptors.</li> <li>4 The Tool produces an optimized format descriptor.</li> <li>5 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A format descriptor has been produced.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.2.2 Interactive Formatting Tools****6.2.2.1 Interactive formatting process**

<b>UCId</b>	UC6.2.2.1
<b>Use case</b>	Interactive formatting process
<b>Description</b>	An Actor wants to format an AXMEDIS Object, interacting with the AXEditor.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.
<b>Assumptions</b>	An AXMEDIS Object has been selected and its digital resources are physically accessible to the Format Tool; profile descriptors for user preferences, device capabilities and delivery context should also be available.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor starts the AXEditor for formatting.</li> <li>2 The Actor performs an interactive template selection.</li> <li>3 The Actor performs an interactive style-sheet selection.</li> <li>4 The Actor performs an interactive style-sheet optimization.</li> <li>5 The Actor performs a format creation.</li> <li>6 The Actor can preview the resulting format applied to the document.</li> <li>7 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A format for the Object has been produced.
<b>Variations</b>	The Actor may stop the process at any step and go back to previous steps.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**6.2.2.2 Interactive template selection**

<b>UCId</b>	UC6.2.2.2
<b>Use case</b>	Interactive template selection
<b>Description</b>	An Actor wants to select a ready-made template for an AXMEDIS Object, interacting with the AXEditor.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.

<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its digital resources are accessible to the Format Tool; profile descriptors for user preferences, device capabilities and delivery context should also be available.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor sends a request for template selection to the AXEditor.</li> <li>2 The Actor provides resource references.</li> <li>3 The Actor may provide Profile references (User, Device and Context Profiles)</li> <li>4 The Actor may give information about the type of document to be created.</li> <li>5 The Actor may give information about the preferred output format.</li> <li>6 The Actor may give information about the target device(s).</li> <li>7 The Actor may give indications about the category of each resource.</li> <li>8 The AXEditor proposes the ordered list of best templates: the Actor can read their description and metadata and choose the best one.</li> <li>9 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A template has been selected.
<b>Variations</b>	The Actor may bypass the selection logic providing a specific template ID.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.2.2.3 Interactive style-sheet selection

<b>UCId</b>	UC6.2.2.3
<b>Use case</b>	Interactive style-sheet selection
<b>Description</b>	An Actor wants to select a ready-made style-sheet, suitable for a given template, interacting with the AXEditor.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its resources are accessible to the Format Tool; a template has also been selected.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor selects a template in the AXEditor.</li> <li>2 The Actor may provide Profile references (User, Device and Context Profiles).</li> <li>3 The Actor may give information about the target device(s).</li> <li>4 The AXEditor proposes the ordered list of best style-sheets suitable for the given template: the Actor can read their description and metadata and choose the best one.</li> <li>5 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	A style-sheet has been selected.
<b>Variations</b>	The Actor may bypass the selection logic providing a specific style-sheet ID.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 6.2.2.4 Interactive style-sheet optimization

<b>UCId</b>	UC6.2.2.4
<b>Use case</b>	Interactive style-sheet optimization
<b>Description</b>	An Actor wants to optimize a style-sheet, interacting with the AXEditor.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.
<b>Assumptions</b>	An AXMEDIS Object has been selected and descriptors for its resources are accessible to the Format Tool; profiles for user preferences, device capabilities and delivery context should be accessible. A template and a style-sheet have also been selected.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor selects a style-sheet in the AXEditor.</li> <li>2 The Actor provides resource references.</li> <li>3 The Actor may provide Profile references (User, Device and Context Profiles).</li> <li>4 The Actor may give information about the target device(s).</li> <li>5 The Tool produces optimized values for parameters defined in the style-sheet and creates a descriptor for the optimized style-sheet.</li> <li>6 The Use Case ends.</li> </ol>
<b>Post-conditions</b>	The style-sheet has been optimized.
<b>Variations</b>	The Actor may manually change some values or provide a new set of values.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.2.2.5 Template creation

<b>UCId</b>	UC6.2.2.5
<b>Use case</b>	Template creation
<b>Description</b>	An Actor wants to create a template for the Format Tool.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.
<b>Assumptions</b>	An AXMEDIS object has been selected and its resources are physically accessible to the AXMEDIS SMIL Editor; otherwise, a set of resource descriptors has to be accessible. In the latter case, “fake” media files are used to preview the format in the Editor.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Use Case begins when the Actor starts the AXMEDIS SMIL Editor.</li> <li>2 The Actor creates a SMIL document which includes the resources contained within the AXMEDIS Object.</li> <li>3 The Actor exports the document as “AXMEDIS Template”.</li> <li>4 The Actor exits from the AXMEDIS SMIL Editor.</li> <li>5 The Use Case ends</li> </ol>
<b>Post-conditions</b>	A SMIL file, that can be used as template for the AXMEDIS Format Engine, is produced.
<b>Variations</b>	<ul style="list-style-type: none"> <li>• The Actor can modify the created template file with a text editor.</li> <li>• The Actor can create a SMIL document with a text editor or an external SMIL editor, open and modify it with the AXMEDIS SMIL editor and export the template.</li> <li>• The Actor can create the template from scratch with a text editor.</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 6.2.2.6 Style-sheet creation

<b>UCId</b>	UC6.2.2.6
<b>Use case</b>	Style-sheet creation
<b>Description</b>	An Actor wants to create a style-sheet for the Format Tool.
<b>Actors</b>	Content owner, Content Integrator, Content Distributor.
<b>Assumptions</b>	An AXMEDIS Object has been selected and its resources are physically accessible to the AXMEDIS SMIL Editor; otherwise, a set of resource descriptors has to be accessible. In the latter case, “fake” media files are used to preview the format in the Editor. Moreover, at least one template for the current document has already been created.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Use Case begins when the Actor starts the AXMEDIS SMIL Editor.</li> <li>2. The Actor creates a SMIL document which includes the resources contained within the AXMEDIS Object.</li> <li>3. The Actor marks some attributes as input for the optimization logic of the Format Tool.</li> <li>4. The Actor chooses a reference template.</li> <li>5. The Actor exports the document as “AXMEDIS Style-sheet”.</li> <li>6. The Actor exits from the AXMEDIS SMIL Editor.</li> <li>7. The Use Case ends</li> </ol>
<b>Post-conditions</b>	An XSLT file, that can be used as style-sheet for an AXMEDIS Template, is produced.
<b>Variations</b>	<ul style="list-style-type: none"> <li>• The Actor can modify the created XSLT file with a text editor.</li> <li>• The Actor can create a SMIL document with a text editor or an external SMIL editor, open and modify it with the AXMEDIS SMIL editor and export the style-sheet.</li> <li>• The Actor can create the style-sheet from scratch with a text editor.</li> <li>• The Actor can create the style-sheet from scratch with an external XSLT editor and modify it to obtain the AXMEDIS-compliance with a text editor.</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 7 AXMEDIS Workflow

### 7.1 Workflow Scenarios

**Assumption:** - The user is already logged in and authenticated by the system. Based on his role, the user is granted a set of “Rights”. The information related to the user (user profile) and the Rights is already available in the AXMEDIS Database.

#### **Scenario 1: - Starting a New Instance of an NPD i.e. New NPD Set-up (A Managerial Task)**

There are times when a user may wish to cause a new workflow process to be set up by “development and configuration technicians” to support new kinds of NPD (New Product Development) with new business process logics. However this scenario relates to occasions when through the Workflow UI, project managers may wish to start a new NPD instance of an already defined workflow process (e.g. the process for producing a new media content, which has been previously defined and configured).

A project manager can thus subsequently assign work activities to individual users or let the assignments to be made automatically by the workflow engine, based on pre-defined rules and roles.

The following scenario describes the process of defining a new NPD within the workflow sub-system. At the end of this scenario, the project manager can expect a fully configured workspace that can be interrogated by users at various levels to give information about all the necessary tasks to be performed, people responsible for performing those tasks, the tools needed to do the tasks, and the location where each task is to be performed, etc, the scenario proceeds as follows:

The user (Manager) is already authenticated and logged into the system.

The user selects the “Add process Definition” button. A new page is loaded to give the identifier for the process. The user enters the details and clicks “Add Process”

The workflow editor/viewer is then launched to enable the user to define the workflow for the new NPD.

The workflow editor launches a blank page (or a page containing the structure of the selected template) for defining the workflow components.

The workflow displays the list of all the processes defined. The user selects the newly created process. The workflow displays the list of activities and transitions.

NPD set-up phase: The user can now select and add components to define the new project. These components can be tasks, people, project, products, objects, places, links, etc. This functionality of the workflow editor is similar to a drawing utility provided by the Microsoft Word editor, which allows the user to add shapes and assign properties to them. However, for the workflow editor it is necessary that all the added components must be connected to at least one other component to form a semantic integration of all the components, which when executed in the defined order produces the required product. Whenever any component is added to the NPD the corresponding properties dialogue box appears for the added component. The user can (re)set the required properties in this dialogue box so as to control the behaviour of the components.

The user can define and add more activities and/or transitions.

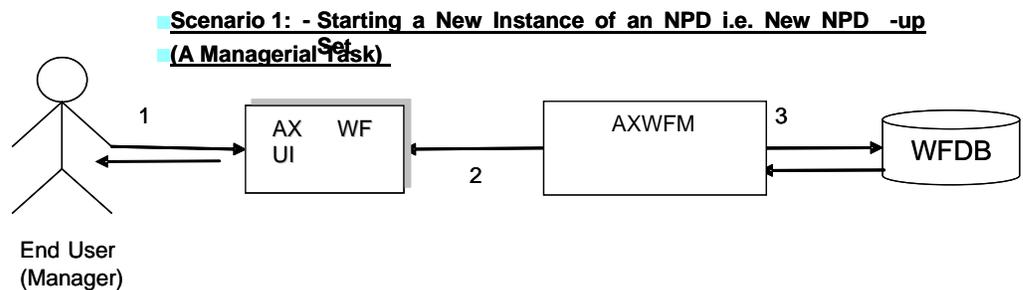
The user can also select the existing workflow process and add as sub-process for newly defined process.

For example the user can add a task to the current NPD workspace and may designate its type as a “Formatting Task”. The user then can add a person to the current workspace and assign his role to be, say, the “Technical Editor” responsible for the formatting Task. He can then link this person to the “Formatting Task”. The user can then add a tool to the current workspace and assign its role as a “Formatting Tool” and then link this tool to the “Formatting Task”. The workflow will interpret these links as “The AXMEDIS Object(ID---) to be formatted with the specific Formatting Tool by the named Technical Editor” thus assigned this task.

It is also possible for the User to define all the tasks and people working on the project first without creating the links. As mentioned before the workflow system can automatically distribute the work to the people, partners, places, etc based on the saved profiles (roles) of the available participating resources and objects.

There are typically two approaches to defining workflow processes: using a specific User Interface or describing the process via a meta-language (e.g. XPD); workflow solutions tend to adopt one or the other of the two approaches).

The command and Reporting is shown in this diagram explicitly as the component that is connected to the editor/viewer to notify termination of the editing and viewing task. In practice, the Command & Reporting module can be viewed as an integral component of AXMEDIS Editor.



**Scenario 2: - Executing Any Task in the Workflow (End-Worker’s Task)**

This scenario describes the process of executing any work Task within the workflow environment. At the end of this scenario, the user can expect the status of the AXMEDIS Object(s) concerned to be updated and the work Task marked as completed thus triggering new sets of tasks as appropriate. This scenario proceeds as follows:

The user (Worker) is already authenticated and logged into the system and the workflow system is up and running. We can therefore assume that the client’s (i.e. session-owner’s) credentials\_ID has been made available to be authenticated for sign-on to initiate the required exchanges with the AXMEDIS tools/engines as service providers and thus the information for authentication has been provided.

The user invokes clicks the “Work list” button for the current workflow selecting a work-item to get the choice of actions to be performed on the work-item for the NPD (or, identically, the workflow-instance) for which he is assigned to perform tasks.

For any selected task, from any given workflow-instance, the Workflow UI displays to the user a choice of available actions and descriptions/suggestions related to the selected work-item (i.e. viewed dynamically these are potential workspace instantiations). These can include actions such as:

EDIT: The user may wish to invoke the AXMEDIS Editor by clicking on Edit and, say, invoke Edit DRM to Edit the DRM of a selected object; this will launch the AXMEDIS DRM Editor.

**SEARCH:** The user may wish to search for all objects involved in a particular NPD, by invoking the Search function of the Workflow UI. The user clicks on Search and then supplies the workflow-instance\_ID. The Workflow Manager passes this query via the Workflow Query Interface through to the Query Support Web Services Interface which submits it to the AXMEDIS Query Support User Interface. This sets up an interaction with the AXMEDIS Object Database to search for all objects involved in the specified process or fulfilling certain criteria.

**SHOW:** The user can request the workflow system to show more information on any selected components (AXMEDIS object(s), tool(s), etc) as may be included in the work-list.

**Terminate Activity:** Users can invoke this functionality to signal to the workflow system their wish to have an activity terminated. Accordingly the workflow system will proceed to the next step in the workflow process instance (It is important to note that this functionality enables an over-ride control action on the part of the human operator if required).

Based on the selected Task the workflow system launches the required tool using the appropriate Interface (e.g. Web-services) or plug-ins associated with that tool. If the tool is in the exclusive access area of the user, the "Check-in" and "Check-out" interfaces will be invoked.

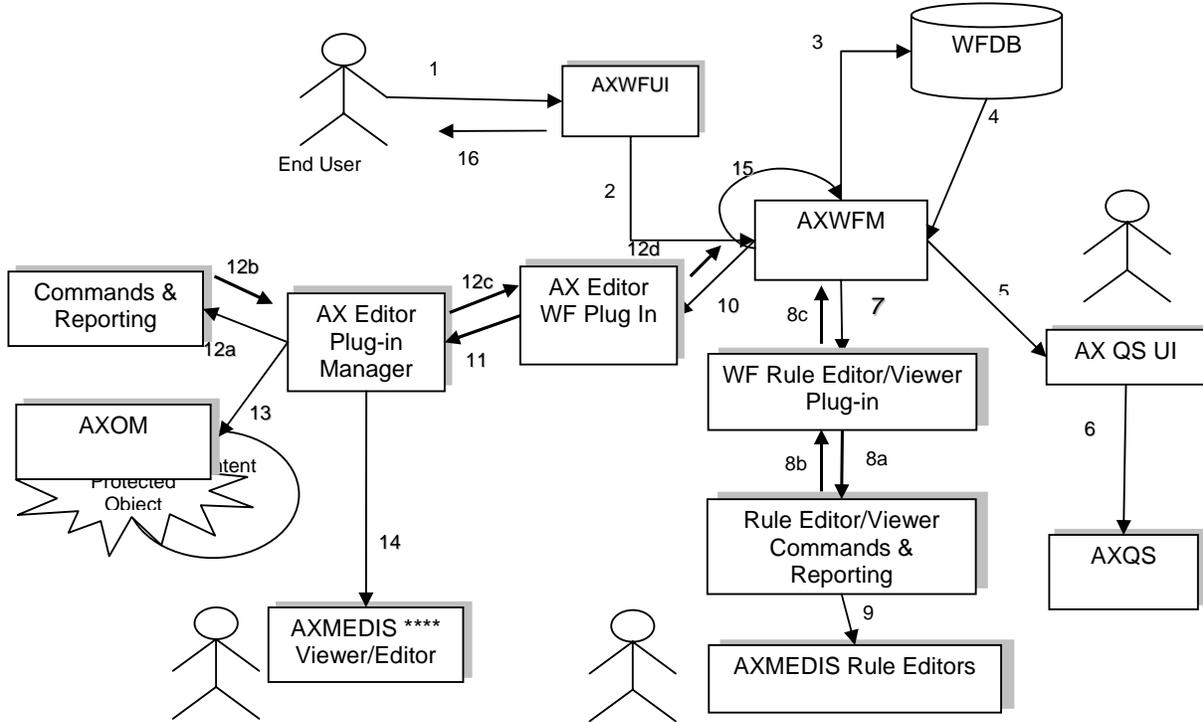
The workflow system assigns a time-stamp to such a Task as the start\_time, which is later referred to while tracking the history of the component.

If required the workflow system will also generate new versions of the AXMEDIS Object. Upon the completion of the Task the workflow system will again assign a time-stamp to this Task as the end time.

At the end of the Task, the workflow system will update the status of the AXMEDIS Object, which may trigger various other tasks (e.g. DRM editing, invoking AXEPTool, etc).

The Command and Reporting is shown in the above scenario diagram explicitly as the component that is connected to the editor/viewer to notify termination of the editing and viewing task. In practice, the Command and Reporting Module can be viewed as an integral component of AXMEDIS Editor.

**Scenario 2: - Executing Any Task in the Workflow (End-Worker's Task)**



The Rule Editor Viewer Command & Reporting module is an integral component of AXMEDIS Editor of the Editor or Engine, while the WF Rule Editor/Viewer Plug in is an external DLL or Plug in module, produced by using the Workflow development tool kit and the AXMEDIS Plug in Development Tool Kit.

**Scenario 3: Sending out Notifications to People**

This scenario describes the process of sending out Notifications initiated by the workflow system or by the people within the workflow environment. At the end of this scenario, the user can expect that notifications are generated and sent to appropriate target(s). The scenario proceeds as follows:

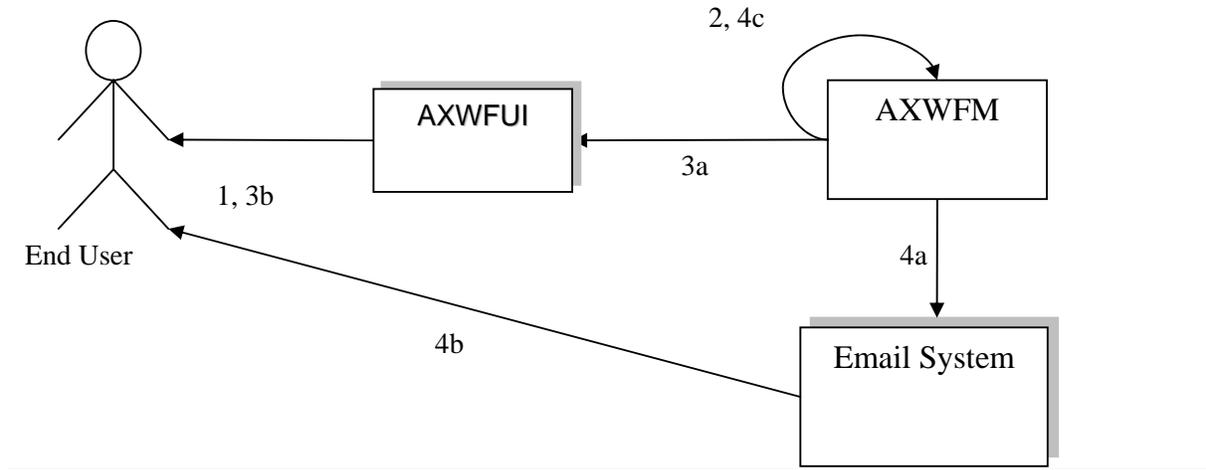
The user (Worker) is already authenticated and logged into the system and the workflow system is up and running. We can thus also assume that the client's (i.e. session-owner's) credentials\_ID has been made available to be authenticated for sign-on to initiate the required exchanges information for authentication and billing purposes.

Upon completion of any Task, the workflow system will generate appropriate Notifications, e.g. If any Task is waiting for the DRM to be cleared, the workflow system will notify this Task by raising the appropriate signal whenever the required DRMs are cleared.

The workflow system can also send out notifications to the users through appropriate tools like e-mailing systems, pop-up messages, etc. e.g. if any actor is waiting for an AXMEDIS object to be downloaded by the AXEPTool, then upon completion of this the Workflow system is notified by the respective Command and Reporting module and it in turn can deliver a pop-up message on the relevant client screen or other designated terminal.

Notifications can also be sent out in the form of e-mails to the user, e.g. if the user has been assigned a new Task, an email will be sent to him regarding this Task and his personal work-list is updated accordingly.

### **Scenario 6: Sending out Notifications to People**



### **Scenario 4: Global View and Tracking of any Component in the Workflow**

This scenario describes the process of generating a global view of any NPD and the tracking of any component within the selected NPD. At the end of this scenario, the user can expect to have the up-to-date progress status of the AXMEDIS Objects within the selected NPD.

The user (Manager/Worker with appropriate rights) is already authenticated and logged into the system and the workflow system is up and running. Therefore we can assume that the client's (i.e. session-owner's) credentials\_ID has been made available to be authenticated for sign-on to initiate the required exchanges information for authentication and billing purposes.

The user selects a particular NPD (or identically a workflow-instance\_ID) and clicks on the Global View icon.

The Workflow system identifies all the components for the selected NPD and launches a set of queries to retrieve information for all of such components from the AXOM through the AXMEDIS Query Support Interface.

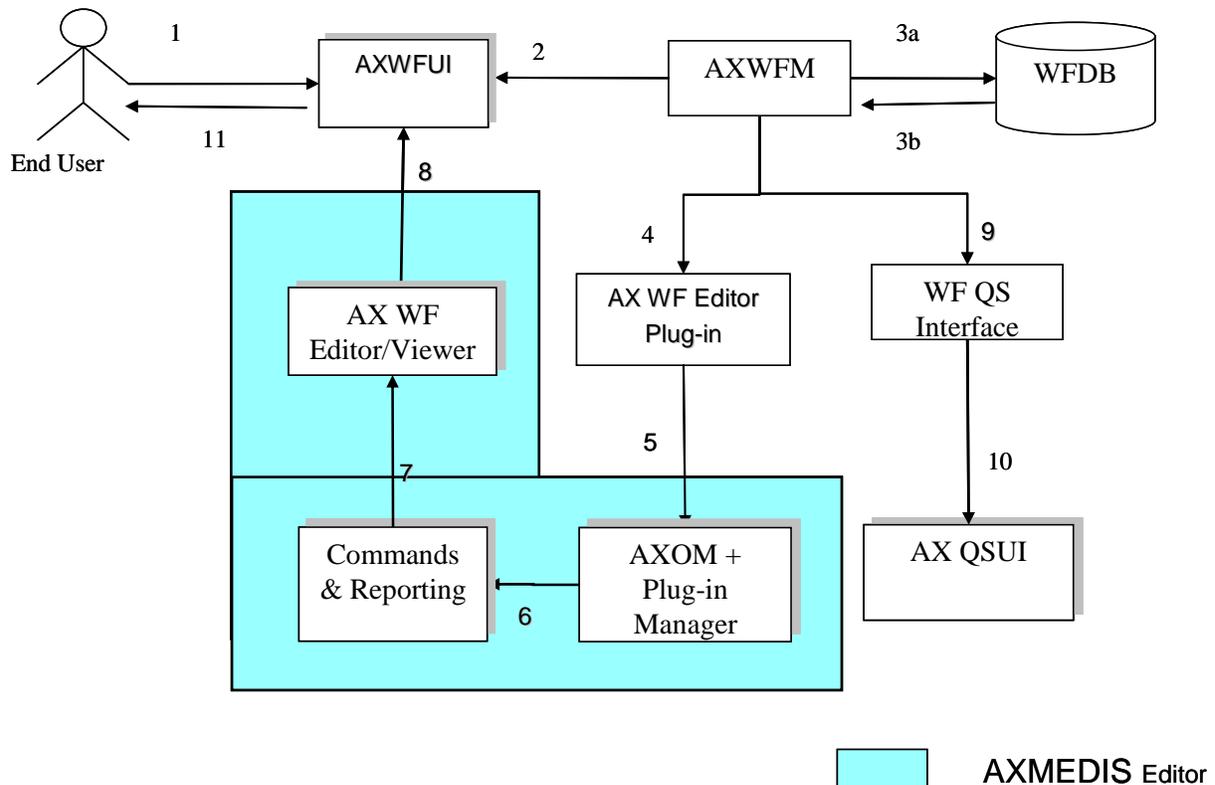
The workflow systems can then launch an Interactive GUI (Workflow viewer) to show the overall status of the NPD workflow along with its Critical Path Tasks (CPA), based on the results received for the above queries.

Through the interactive GUI, the user can select any individual component and can demand more information on it. This component can be any object, task, person, etc.

Accordingly the workflow system can launch a responsive query to retrieve detailed information regarding the component(s) selected by the user.

The command and Reporting is shown in the scenario diagram explicitly as the component that is connected to the editor/viewer to notify termination of the editing and viewing task. In practice, the Command & Reporting module can be viewed as an integral component of AXMEDIS Editor.

**Scenario 7: Global View and Tracking of any Component in the Workflow**



The Command & Reporting module is an integral component of AXMEDIS Editor, while the AXMEDIS Editor WF Plug in is an external DLL or Plug in module, produced by using the Workflow development tool kit and the AXMEDIS Plug in Development Tool Kit.

**Scenario 5: Invoking the Composition and Formatting Engine**

This scenario describes the interaction between the workflow and the Composition and Formatting Engine to compose/format any AXMEDIS Object according to selected composition/formatting rules (Rule-ID).

The control of all the composition/formating engine and AXEPTool will now be through AXCP engine. For this various rules will be loaded in the AXCP engine and executed accordingly.

It is assumed that the composition and/or formatting task(s) can be carried out autonomously, without the intervention of the user but it can be done on an adhoc basis synchronously at user’s instant request. In any event we can also assume that the client’s (i.e. project-owner’s) credentials\_ID has been made available to be authenticated for sign-on to initiate the required exchanges information for authentication and billing purposes.

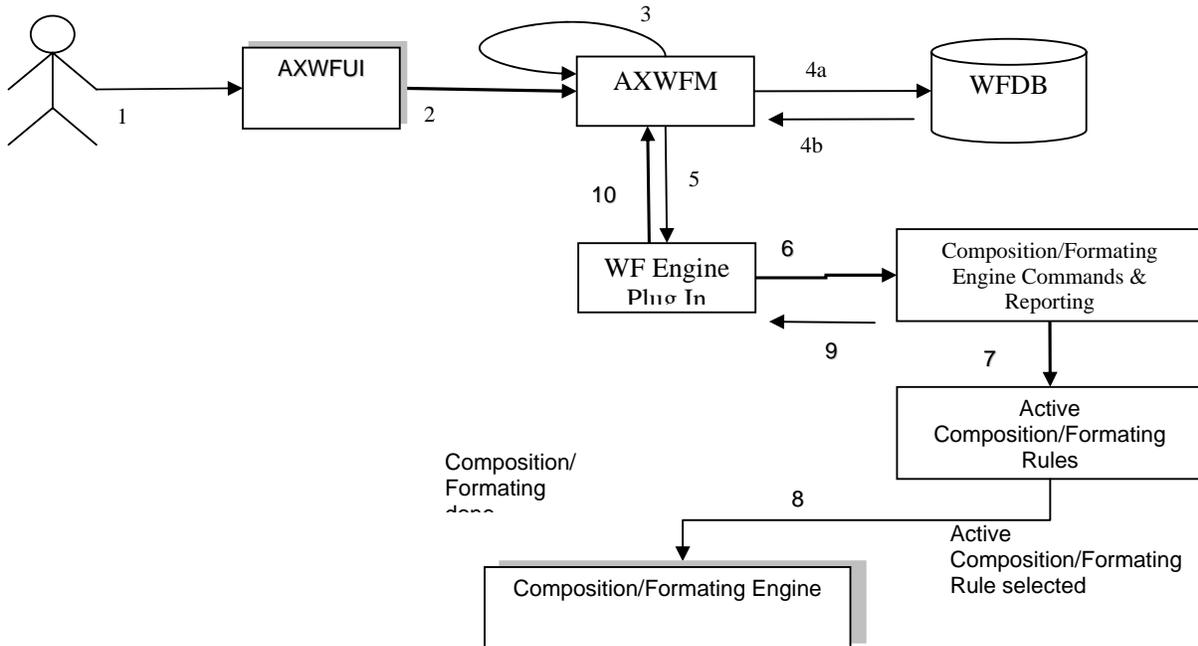
The workflow system is up and running.

The workflow system effects the request to the Composition/Formatting engine via the Workflow Plug-in linking through the Command & Reporting Module through to the Composition and Formatting Active Rules module. In this way the workflow system passes to the Composition and Formatting engine an Activate compose/format request together with a composition/Formatting Rule ID and Object ID to control the correct composition/formatting of the right object(s) from the Active List.

The Composition/Formatting Engine then composes/formats the relevant AXMEDIS Object(s) as required per specified (or default) composition/formatting rules

Upon completion of the composition/formatting, the WFMS is informed by the Command & Reporting Module and the metadata of the relevant Object is also updated accordingly.

**Scenario 8: Invoking the Composition/Formatting Engine**



**7.2 Controlling and supervising local AXMEDIS tools**

**7.2.1 General WorkFlow Use Cases**

**7.2.1.1 Create NPD Workspace**

<b>UCId</b>	UC7.2.1.1
<b>Use case</b>	Create a NPD
<b>Description</b>	This use case when run should create a fresh NPD workspace folder with the required configuration files in it etc i.e. a suitable workspace desktop suited to the role of the participant(s) in the value chain segment to which they are contributing towards the NPD as a whole
<b>Actors</b>	Creator
<b>Assumptions</b>	Always valid: user has been identified by System; User has the correct rights
<b>Steps</b>	1 Actor chooses "Create NPD".
<b>Post-conditions</b>	New NPD project(s) space created in the user client & P2P desktops New NPD creation process instance started
<b>Variations</b>	Actor has no rights
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**7.2.1.2 Add components to the NPD**

<b>UCId</b>	UC7.2.1.2
<b>Use case</b>	Add components to the NPD
<b>Description</b>	This use case is responsible for adding components to the NPD. Typically it can be inherited to add projects, people, roles, processes, phases, partners, components, activities, Rights, DRM, etc
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor clicks on “Add component” button. 2 A template is opened listing various types of Workflow Components 3 The user selects the desired component to be added 4 The user sets the desired properties of the selected component
<b>Post-conditions</b>	New component added to active NPD. Started (if any) a subprocess for managing the newly created object
<b>Variations</b>	Actor has no rights Component and AXMEDIS Object incompatibility
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.3 Edit information of the NPD

<b>UCId</b>	UC7.2.1.3
<b>Use case</b>	Edit information of the NPD Note: this is a Use case with Workflow tight integration to editors (multiple interface)
<b>Description</b>	This use case is responsible for editing various aspects of the NPD. It can be used to edit the current DRM rules or can be used to edit a component based on the selected process and updates versions if required.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System; NPD must exist; Actor has the correct rights.
<b>Steps</b>	1 Actor clicks on “Edit” button. 2 The property dialogue box opens asking user, the necessary actions for editing. 3 Based on the action selected the Workflow launches appropriate editor/tool.
<b>Post-conditions</b>	Proper editor invoked for active NPD.
<b>Variations</b>	Actor has no rights
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.4 Delete information of a NPD

<b>UCId</b>	UC7.2.1.4
<b>Use case</b>	Remove information of a NPD
<b>Description</b>	This is a generic use case responsible for removing anything from the NPD. e.g. partners, people, processes, components, etc.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator

<b>Assumptions</b>	An non-empty NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor selects component to remove then Actor clicks on 'remove'. 2 Optional confirmation dialogue.
<b>Post-conditions</b>	Selected component deleted from active NPD.
<b>Variations</b>	Actor has no rights
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.5 Show information regarding components of a NPD

<b>UCId</b>	UC7.2.1.5
<b>Use case</b>	Show information regarding components of a NPD
<b>Description</b>	This use case is responsible for showing information related to various components, their copyrights, DRM/PAR, History (metadata, timestamp, version), Template (house styles, business rules), global state of any projects, etc.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System.
<b>Steps</b>	1 Actor clicks on "Show". 2 An appropriate viewer is launched to show the requested details.
<b>Post-conditions</b>	Properties related to the active NPD displayed.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.6 Delete a NPD

<b>UCId</b>	UC7.2.1.6
<b>Use case</b>	Delete NPD
<b>Description</b>	This destroys the NPD workspace, when the decision of No-Go is taken. This removes all the information regarding the NPD.
<b>Actors</b>	Creator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor clicks on "Discard NPD". 2 Confirmation dialogue.
<b>Post-conditions</b>	Active NPD deleted along with associated components. The process instance initiated with the NPD instance creation is aborted.
<b>Variations</b>	No rights.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.7 Search a NPD

<b>UCId</b>	UC7.2.1.7
<b>Use case</b>	Search a NPD

<b>Description</b>	This is a generic use case that can search for a NPD. A special case can be inherited to search for eligible components to be worked on.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System.
<b>Steps</b>	1 Actor clicks on “Search” button. 2 A query dialogue box appears 3 Actor inserts the required parameters and launch the search
<b>Post-conditions</b>	The result of performed search is displayed in the form of a list
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.8 Track Component

<b>UCId</b>	UC7.2.1.8
<b>Use case</b>	Track component
<b>Description</b>	This tracks down the history of the selected component. The result comprises of all the actions performed on the component along with all the future activities including “wait actions” re “suspended” objects awaiting pending operations which may themselves be contingent on Critical Path Action(s) (CPA) trigger(s).
<b>Actors</b>	Creator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System.
<b>Steps</b>	1 Actor selects a component. 2 Actor clicks on “Track component” button.
<b>Post-conditions</b>	History and planned steps of selected component displayed.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.9 Timestamp Generator

<b>UCId</b>	UC7.2.1.9
<b>Use case</b>	Timestamp generator
<b>Description</b>	This use case is responsible for generating the timestamp for each of the activities that are performed on an object by an actor or process at anytime, anywhere any place by any partner – in any phase of the production and distribution end-to-end. This can be represented within the metadata and will be used by “Track Component” to locate the evolution status of any object within nested spiral development lifecycles across distributed teams from different units/partners. This will allow global tracking including accommodating re-entrant and re-cursive states of processing of the objects across partner project spaces (projects, phases, processes, persons, partners, places, periods, purpose, progress-to-date, project-work-remaining – 10P STAMP, Badii 2004)
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	A non-empty NPD must be active/open
<b>Steps</b>	1 The workflow manager will log the beginning and end of any task performed on any object.

<b>Post-conditions</b>	An appropriate timestamp is associated with the objects.
<b>Variations</b>	This use case can be tested as expected result for each of the other cases.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 7.2.1.10 List of Work

<b>UCId</b>	UC7.2.1.10
<b>Use case</b>	List Work
<b>Description</b>	This use case is responsible for generating a hierarchical list of the sequence of all the work to be done in a particular sectorial workflow scenario, e.g. phases, processes to be invoked on certain objects by certain people with specific globally traceable coordinates as unique and easily retrievable instances (i.e. 10P Stamped Workflow Objects).
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	Actor has been identified by System; There are work items to which the actor is assigned
<b>Steps</b>	1 Actor clicks on “show work list” button. 2 The actor can interact with the list of works
<b>Post-conditions</b>	The actor work list is displayed
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 7.2.1.11 Select a Work Item from the List of Work

<b>UCId</b>	UC7.2.1.11
<b>Use case</b>	Select a Work Item from the List of Work
<b>Description</b>	This use case is responsible for selecting a work item from the work list
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	Actor has been identified by System The actor has executed the “personal work list” case or “list work” case; There are work items to which the actor is assigned
<b>Steps</b>	1 Actor clicks on “select work item” button. 2 Once selected the actor can do some editing activities
<b>Post-conditions</b>	The actor work item activity list and/or description is displayed
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 7.2.1.12 Complete a task of a work Item

<b>UCId</b>	UC7.2.1.12
<b>Use case</b>	Complete a task of a work Item
<b>Description</b>	Users can invoke this functionality to signal to the workflow system their wish to have an activity terminated. Accordingly the workflow system will proceed to the next step in the workflow process instance (It is important to note that this functionality enables an over-ride control action on the part of the human operator if required)

<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	Actor has been identified by System The actor has previously selected a work item from the work list. The actor has completed the activity related to work item
<b>Steps</b>	1 Actor clicks on “terminate work item task” button.
<b>Post-conditions</b>	The actor work item is completed, it is deleted from the user’s work list and the Workflow passes to the next activity as planned for the process instance flow
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.13 Change State/Phase of a Task for a work Item

<b>UCId</b>	UC7.2.1.13
<b>Use case</b>	Change State/Phase of a Task for a work Item
<b>Description</b>	This use case is responsible for changing states of objects/actors or phases of a project including triggering and the upload of a new workspace for a new phase in the project, e.g. the object may become available after copy right clearance or a person/partner may become (un)available.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor selects a component 2 Actor then click “change state”.
<b>Post-conditions</b>	State/phase is changed for the selected component or actor. The changing of the state/phase will then trigger a set of sub-sequent activities that are necessary to be performed before the start of new Phase.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.2.1.14 Notification of information to a personnel for a task of a work

<b>UCId</b>	UC7.2.1.14
<b>Use case</b>	Notification of information to a personnel for a task of a work
<b>Description</b>	This use case is responsible for sending out notifications to the responsible actors for the start and/or end of the activities/work; e.g. request for information or components, etc.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An non-empty NPD must be active/open
<b>Steps</b>	1 Actor selects one or more actors, select from list of message types then clicks “notify”.
<b>Post-conditions</b>	Appropriate notification is sent to responsible actors via appropriate tool (e.g. email).
<b>Variations</b>	The actor may wish to type his own personal message.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**7.2.1.15 Global Viewer of all information of a NPD**

<b>UCId</b>	UC7.2.1.15
<b>Use case</b>	Global viewer of all information of a NPD
<b>Description</b>	This use case is to collect all the information for the current NPD and present a global view for managerial decisions and for Production accounting information feed made accessible any Enterprise MIS platforms such as SAP (along with the 10P Object Stamps)
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor selects “global view”.
<b>Post-conditions</b>	Global information is displayed/exported for the active NPD.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**7.2.1.16 Check-in task performed by manual operator**

<b>UCId</b>	UC7.2.1.16
<b>Use case</b>	Check-in task performed by manual operator Note: this is a Use case with Workflow loose integration to editors (simple interface)
<b>Description</b>	This use case is responsible for editing manually various aspects of the NPD. It can be used to edit the current DRM rules or can be used to edit a component based on the selected process and updates versions if required.
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System
<b>Steps</b>	1 Actor clicks on “check-in” button.
<b>Post-conditions</b>	The object is copied from AXMEDIS DB To an area for exclusive access of the actor, ready to be downloaded
<b>Variations</b>	Actor has no rights
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**7.2.1.17 Check-out task performed by manual operator**

<b>UCId</b>	UC7.2.1.17
<b>Use case</b>	Check-out task performed by manual operator Note: this is a Use case with Workflow loose integration to editors (simple interface)
<b>Description</b>	This use case is responsible for copying the object from the actor exclusive access area (when he previously uploaded it) to the AXMEDIS DB
<b>Actors</b>	Creator, Producer, Integrator, Aggregator
<b>Assumptions</b>	An NPD process instance is active, a task was assigned to the actor, the actor selected a work item Actor has been identified by System; Actor has previously checked-in.
<b>Steps</b>	1 Actor clicks on “check-out” button.
<b>Post-conditions</b>	The file is copied in the AXMEDIS DB

<b>Variations</b>	Actor has no rights It can automatically execute the “task completed”
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 7.3 Controlling and Supervising AXMEDIS Object life in AXMEDIS compliant factories

The AXMEDIS Object Manager will be expected to satisfy the demands of three categories of customers in general:

- 1) Producer-Consumers (Prosumers)
- 2) Consumers
- 3) DRM/Licensing Management Authorisers/Supervisors

The modes of interaction with the above three types of actors that may lead to some work to be done on Objects and the tracking and control of such work by the AXMEDIS Object Workflow Manager is expected to include the following scenarios:

**A)** When Prosumers act as producers of digital objects being contributed by them as AXMEDIS Objects and therefore wish to add a new (“invented”) Object to the AXMEDIS Objects.

**B)** When Prosumer act as consumers and when they want to take an AXMEDIS Object. This may require the triggering of authorisation/ licensing/protection tools and the relevant access/updates.

**C)** When either the prosumers or pure consumers register an order for a particular AXMEDIS Object to be made available in a particular state that may not be readily available at the first Query instant (i.e. the Object is needed but in a particular view, with a certain rendering , with a specific embedded element modified/added/subtracted), then some Editor/Viewer or DRM Editing function (in turn itself invoking the Protection tool) or other plug-ins may need to be triggered so that the Object is thus (re)worked and turned from being “not-ready” to being “ready” for the customer’s instance of “usage”, then licensing management can be triggered before the “usage” instance is allowed to proceed.

To enable any necessary actions to be done on an AXMEDIS Object to satisfy all kinds of users’ demands, presently, the responsibility boundary between the AXMEDIS Editor and the AXMEDIS Workflow Objects Manager is not fully established in terms of which is to be triggering which tools/plug-ins.

However the AXMEDIS Workflow Object Manager has to be able to track every operation that needs to be performed on any Object. It can use a PIA in order to access the status of Objects modified by the AXMEDIS Editors/Viewers or by any other plug-ins and to trigger the editor or other tools in order to invoke and track such further modifications as may be necessary on any object so as to satisfy customer demands; these can include:

**a) View/update** Object metadata

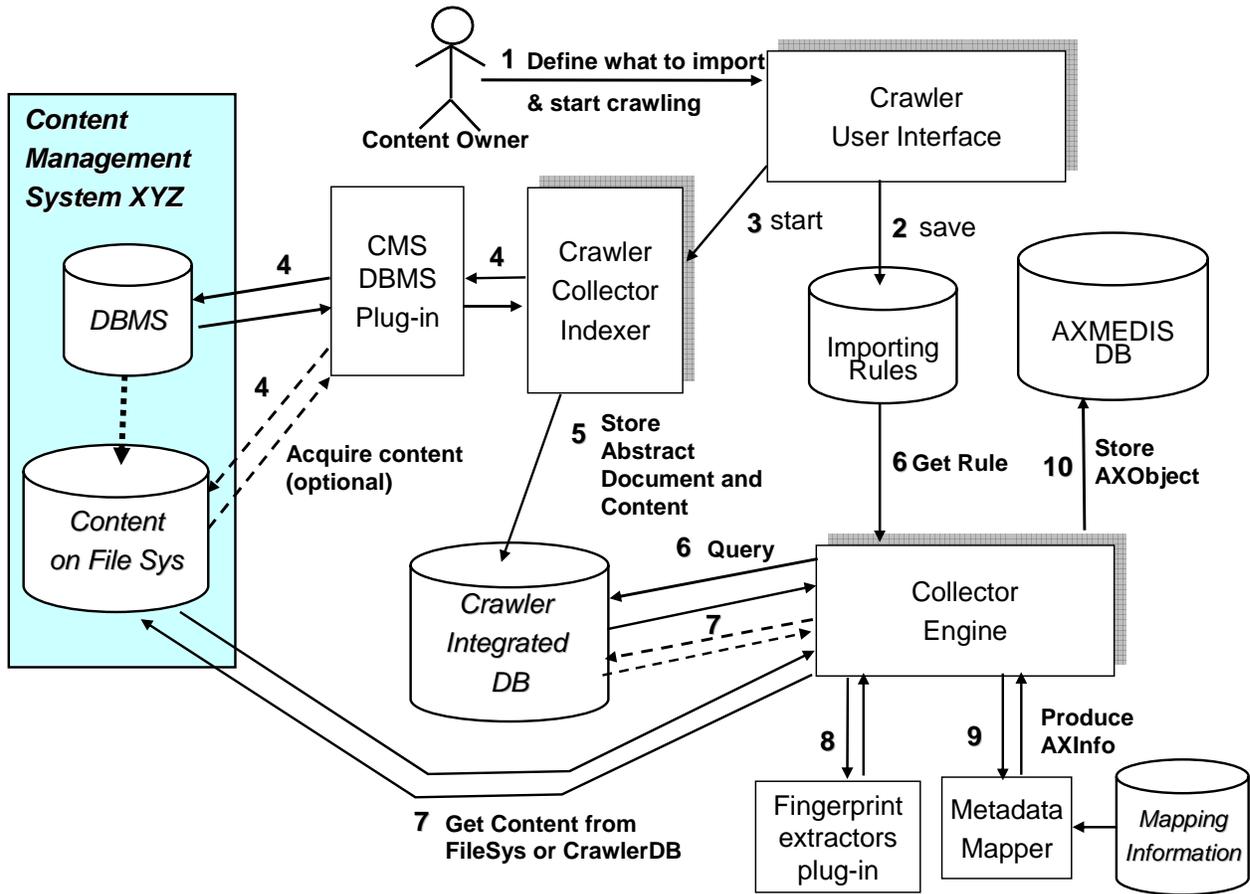
**b) Create/modify** objects or object-parts (e.g. new required views, new rendering, new modifications on embedded element(s) etc.)

**c) Invoke the AXMEDIS Protection Tool and/or Licensing Manager/ Content Processing** -triggering a (meta-data)-Editor/Viewer or other plug-ins. The Protection Tool is the AXMEDIS tool which controls all interactions with the AXMEDIS Object Manager and all other AXMEDIS Editor/Viewers and plug-ins to guarantee that both the users’ granted rights and the owners’ protections are respected).

**AXMEDIS Object Workflow Manager Triggers:** The following comprise the pool of trigger types initially encountered in the relevant domain analysis: new-usage-instance-needed, new action, full-rights, relative-rights, right-updated, rights-granted, rights-denied, protection-status, unprotected, protected, modified/rendered, new-views-created, interrupted-process-n, history, metadata-updated, , metadata-viewed, metadata missing/incomplete, ready, not-ready, stopped, formatted, packaged, edit-started, edit-completed, protection-tool-started, protection-tool-ended, license-manager-started, license-manager-ended, wanted, deposited, owned, viewed-n, taken-n, requested-n, time-done, phase-done, process-done, waiting-on/for-process-n, awaited-by-process-n, suspended, internal, external, authorised/signed-off

These will be the subject of further analysis for the next version of this Requirements document.

## 8 AXMEDIS Object Acquisition from CMS



### 8.1 Automatic gathering of Content, Collector Engine

#### 8.1.1 Setup for metadata mapping

<b>UCId</b>	UC8.1.1
<b>Use case</b>	Setup for metadata mapping
<b>Description</b>	An Actor wants to setup the environment for CMS crawling.
<b>Actors</b>	Content Provider, Content Distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor runs the metadatamapper application</li> <li>2 An example/model of the source and destination metadata is loaded (it can be based on a couple of objects)</li> <li>3 The actor associates elements in the source metadata with elements in the destination metadata</li> <li>4 The actor generates a map file which can be used to convert source xml metadata into destination xml metadata</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	An XSL may be used for this task
<b>Issues</b>	None

#### 8.1.2 Setup for content crawling

<b>UCId</b>	UC8.1.2
<b>Use case</b>	Setup for content crawling
<b>Description</b>	An Actor wants to setup the environment for CMS crawling.
<b>Actors</b>	Content Provider, Content Distributor
<b>Assumptions</b>	Content crawling environment was set up.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, interacts with the Crawler Collector Indexer to add/remove/modify an crawling rule</li> <li>2 When adding/modifying a crawling rule the Actor states: <ol style="list-style-type: none"> <li>2.1 the source to crawl and all the needed information</li> <li>2.2 possibly which fingerprints extractor has to be run on the content</li> <li>2.3 possibly which adaptation algorithm to use</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 8.1.3 Define what content to acquire from Crawled Integrated Database

<b>UCId</b>	UC8.1.3
<b>Use case</b>	Define what content to acquire from Crawled Integrated Database
<b>Description</b>	An Actor wants to setup the environment for CMS gatering.
<b>Actors</b>	Content Provider, Content Distributor
<b>Assumptions</b>	Content crawling environment was set up.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, interacts with the Collector Engine User Interface to add/remove/modify an importing rule.</li> <li>2 When adding/modifying a rule the Actor states: <ol style="list-style-type: none"> <li>2.1 which properties has to have the content to be imported (e.g. all audio files of Ramazzotti, later that 1990)</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

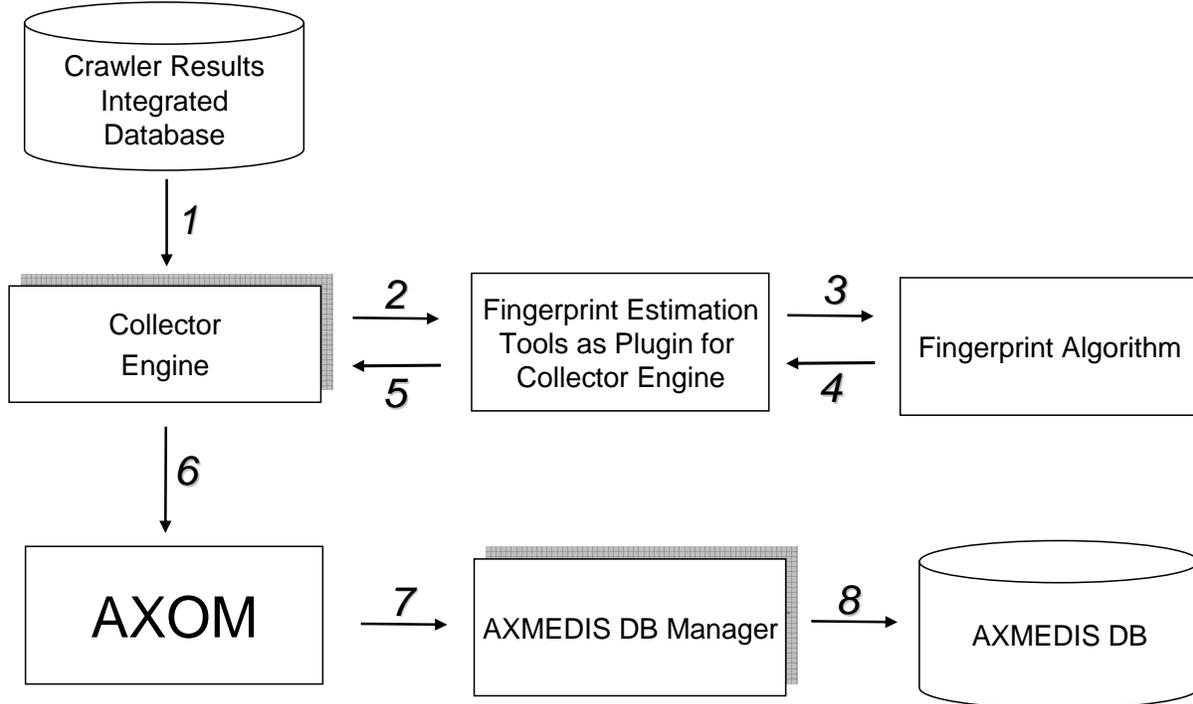
### 8.1.4 Start content crawling

<b>UCId</b>	UC8.1.4
<b>Use case</b>	Start content crawling
<b>Description</b>	An Actor wants to start CMS crawling.
<b>Actors</b>	Content Provider, Content Distributor
<b>Assumptions</b>	The metadata mapping and the crawler have been set up.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, starts CMS crawling using Collector Engine User Interface</li> <li>2 The Crawler Collector Indexer starts accessing to the CMS database using a specific plug-in, it collects the data from the DB and it stores the data into the Crawler Integrated Database. <ol style="list-style-type: none"> <li>2.1 the Collector Engine accesses to the new/updated content and checks the import rules</li> <li>2.2 If the content has to be imported <ol style="list-style-type: none"> <li>2.2.1 the content is fingerprinted and/or adapted (using plug-ins) as instructed by the import rule</li> <li>2.2.2 the AXMEDIS Object is built using the metadata coming from the Crawler Database</li> <li>2.2.3 the AXMEDIS Object is uploaded in the AXMEDIS Database</li> </ol> </li> </ol> </li> </ol>
<b>Post-conditions</b>	None

<b>Variations</b>	None
<b>Asynchronous actions</b>	Content Acquisition can be stopped
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 8.2 Fingerprint Extractor as a collection of Collector Engine Plug-ins for extracting features

### 8.2.1 Calculating content descriptors/fingerprint during crawling



<b>UCId</b>	UC7.2.1
<b>Use case</b>	Calculating content descriptors/fingerprint during crawling
<b>Description</b>	After collecting content by the crawler collector indexer the crawler collector indexer initiates the calculation of the fingerprint.
<b>Actors</b>	Producer
<b>Assumptions</b>	Content was indexed by Crawler Collector Indexer
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Fingerprinting methods are called by the Rule Engine in the AXCP on the script</li> <li>2 Content descriptors and fingerprint/descriptors values are calculated.</li> <li>3 Content descriptors and fingerprint/descriptors values are stored in the body of the objects as metadata descriptors. Some of them can be used for indexing the database when the object is inserted into the database</li> <li>4 The object is loaded into the database or saved in the disk</li> </ol>
<b>Post-conditions</b>	Content descriptors and fingerprints are stored in the database and into the AXMEDIS object structure to save them into the AXOB.
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Different kinds of media types</li> <li>• Compound objects</li> </ul>
<b>Asynchronous actions</b>	

<b>Design suggestions</b>	The structure of the content descriptors influences the database retrieval. Especially in case of processed and modified content the database query must not be limited to exact matches but also consider similarity search (e.g. nearest neighbour).
<b>Issues</b>	In general, content descriptors are (meta) information that is extracted automatically. Two different kinds of descriptors can be identified: low-level and high-level descriptors. While high-level descriptors have a semantic meaning (directly understandable by humans like the melody) low-level descriptors are haven't. Fingerprints are a special kind of content descriptors. Although they uniquely identify content they have to be distinguished from OID. The OID is a consecutive number generated univocally buy the AXCS. Fingerprints are automatically calculated codes and also called perceptual hashed: They can be used to uniquely identify the digital resources, and may be the whole object as an extreme case. Within AXMEDIS also cryptographic hash functions have to be considered.

## 9 AXMEDIS Database

### 9.1 Managing a Database of AXMEDIS Objects

#### 9.1.1 Administer Objects in the AXMEDIS DB

<b>UCId</b>	UC9.1.1
<b>Use case</b>	Administer Objects in the AXMEDIS DB
<b>Description</b>	An Actor performs administrative tasks on AXMEDIS DB related to objects.
<b>Actors</b>	Content Integrator, Content Distributor, and in general all the user that have an AXMEDIS DB in-house
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Actor perform an administrative operation on the AXMEDIS DB such has: query, browse, open an object, remove a version of object, remove an object</li> <li>2. The AXMEDIS object database perform the operations</li> <li>3. The actor is notified about the success of the operation and the result of the operation id transmitted back to the actor.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 9.1.2 Administer User in the AXMEDIS DB

<b>UCId</b>	UC9.1.2
<b>Use case</b>	Administer Objects in the AXMEDIS DB
<b>Description</b>	An Actor perform administrative tasks on AXMEDIS DB related to users.
<b>Actors</b>	Content Integrator, Content Distributor, and in general all the user that have an AXMEDIS DB in-house
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor perform an administrative operation on the AXMEDIS DB such has: add/remove/modify a user of modifying the grant of the user</li> <li>2 The AXMEDIS object database perform the operations</li> <li>3 The actor is notified about the success of the operation and the result of the operation id transmitted back to the Actor.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 9.1.3 Accessing a specific version of an AXMEDIS object

<b>UCId</b>	UC9.1.3
<b>Use case</b>	Accessing a specific version of an AXMEDIS object
<b>Description</b>	An Actor wants to see a specific version of an object interacting with the AXMEDIS Database Administration Tool.
<b>Actors</b>	Content Integrator, Content Distributor, Content Owner
<b>Assumptions</b>	None

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, login into the system specifying the username and a password, if not already done</li> <li>2 The Actor makes a query or browses the database looking for the object</li> <li>3 The Actor selects to see the history of the object</li> <li>4 The Tool reports the history containing the date of upload</li> <li>5 The Actor selects the version to view and opens it using the AXMEDIS Editor</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	When a specific version is opened using the AXMEDIS Editor, AXMEDIS Editor cannot upload it again in the database

#### 9.1.4 Removing last version of an AXMEDIS object

<b>UCId</b>	UC9.1.4
<b>Use case</b>	Removing last version of an AXMEDIS object
<b>Description</b>	An Actor wants to remove the last version of a specific object interacting with the AXMEDIS Database Administration Tool.
<b>Actors</b>	Content Integrator, Content Distributor, Content Owner
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, login into the system specifying the username and a password, if not already done</li> <li>2 The Actor makes a query or browses the database looking for the object</li> <li>3 The Actor tries to perform the removal of the last version of the object, if the Actor has the grant to perform the removal of last version of an object the action succeed, it fails otherwise.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<p>The Actor may perform a query and remove the last version of all the objects resulting from the query or for a subset of them. In case the removal of one of them fails the tool should ask to continue or to stop.</p> <p>The Actor may ask to remove versions to reconstruct the state of the object/objects up to a specific date.</p>
<b>Asynchronous actions</b>	In case of multiple removal the Actor can stop the process.
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 9.1.5 Removing an AXMEDIS object

<b>UCId</b>	UC9.1.5
<b>Use case</b>	Removing an AXMEDIS object
<b>Description</b>	An Actor wants to remove a specific object interacting with the AXMEDIS Database Administration Tool.
<b>Actors</b>	Content Integrator, Content Distributor, Content Owner
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, login into the system specifying the username and a password, if not already done</li> <li>2 The Actor makes a query or browses the database looking for the object</li> <li>3 The Actor tries to perform the removal of the object, if the Actor has the grant to perform the removal of an object the action succeed, it fails otherwise.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	The Actor may perform a query and remove all the objects resulting from the query or a subset of them. In case the removal of one of them fails the tool should ask to continue or to stop.

<b>Asynchronous actions</b>	In case of multiple removal the Actor can stop the process.
<b>Design suggestions</b>	None
<b>Issues</b>	If transactions have been performed for the object the operation may fail

### 9.1.6 User Management

<b>UCId</b>	UC9.1.6
<b>Use case</b>	User Management
<b>Description</b>	An Actor wants to add/remove/modify user information to access to the database
<b>Actors</b>	Content Integrator, Content Distributor, Content Owner
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, login into the system specifying the username and a password, if not already done</li> <li>2 The Actor selects the user management tool</li> <li>3 The Tool presents the list of users having access to the system (if the user has the user management grant)</li> <li>4 The Actor can: <ol style="list-style-type: none"> <li>4.1 Add a new user giving: the username, a password, e-mail, description, the groups he/she belongs to, the grants</li> <li>4.2 Remove an user</li> <li>4.3 Modify the user data</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	The OS or DBMS users management support may be used.
<b>Issues</b>	Users should be kept synchronized with the workflow users

### 9.1.7 User Groups Management

<b>UCId</b>	UC9.1.7
<b>Use case</b>	User Groups Management
<b>Description</b>	An Actor wants to add/remove/modify groups of users
<b>Actors</b>	Content Integrator, Content Distributor, Content Owner
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, login into the system specifying the username and a password, if not already done</li> <li>2 The Actor selects the group management tool</li> <li>3 The Tool presents the list of groups having access to the system (if the user has the user management grant) and for each group the users belonging to it</li> <li>4 The Actor can: <ol style="list-style-type: none"> <li>4.1 Add a new group</li> <li>4.2 Remove a group (if no users are in it)</li> <li>4.3 Modify the group</li> <li>4.4 Add/remove users from the groups</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	The OS or DBMS users management support may be used.
<b>Issues</b>	Users should be kept synchronized with the workflow users

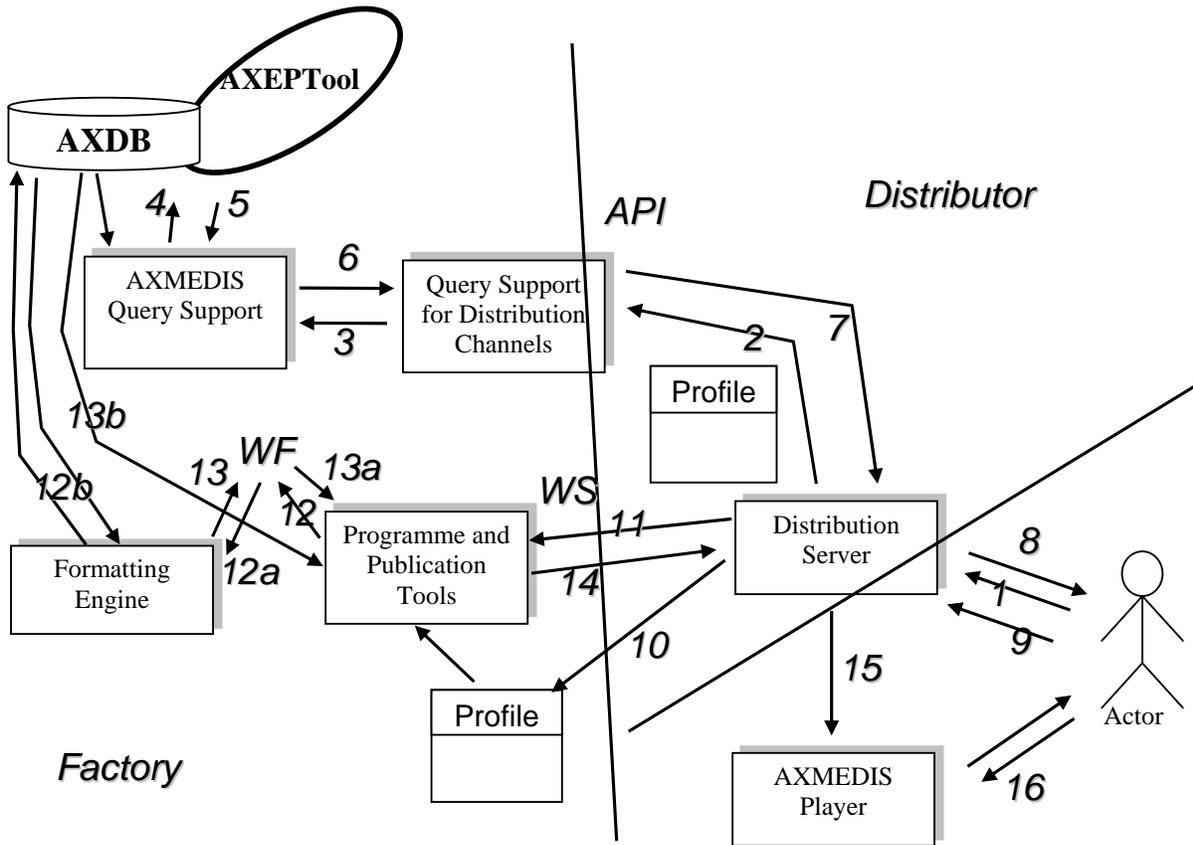
## 9.2 Making queries inside Databases of AXMEDIS objects and inside the objects

Selection creation is explained in the general use case section

### 9.2.1 Querying for AXMEDIS objects and inside objects

<b>UCId</b>	UC9.2.1
<b>Use case</b>	Querying for AXMEDIS objects and inside objects
<b>Description</b>	An Actor is looking for an AXMEDIS object or a set of AXMEDIS objects which satisfy a set of technical, right or feature related conditions.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, composes a query on the aspects of interest (technical, DRM or feature related). Furthermore, the Actor chooses “where” to search for available AXMEDIS objects: within local AXMEDIS Database, within an AXMEDIS object, on AXEPTool network or among those contents which have to be collected, by the Collector Engine, and have not yet.</li> <li>2 The Actor submits the queries previously composed to execute the search</li> <li>3 AXMEDIS Query Support User Interface, using the AXMEDIS Query Support, submits the Actor’s query to each of the chosen search “places” by using the corresponding specific interface: (i) Collector Engine Query Support Interface, (ii) AXEPTool Query Support Interface and (iii) AXMEDIS Database Manager (iv) AXMEDIS Data Model Query Support</li> <li>4 AXMEDIS Query Support merges the results all together and return the complete list to the AXMEDIS Query Support User Interface</li> <li>5 AXMEDIS Query Support User Interface shows the result to the Actor in an adequate manner, i.e. in such a way that the Actor can understand: (i) from which source an object come (ii) which are the restriction on the object (iii) etc...</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Web services should be adopted at the communication layer
<b>Issues</b>	None

### 9.2.2 Querying for AXMEDIS from Clients



<b>UCId</b>	UC9.2.2
<b>Use case</b>	Querying for AXMEDIS From Clients
<b>Description</b>	An Actor (end user) is looking for an AXMEDIS object or a set of AXMEDIS objects which satisfy a set of technical, right or feature related conditions.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	None

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, composes a query on the aspects of interest (technical, DRM or feature related). Furthermore, the Actor chooses “where” to search for available AXMEDIS objects: within local AXMEDIS Database, within an AXMEDIS object, on AXEPTool network or among those contents which have to be collected, by the Collector Engine, and have not yet.</li> <li>2 The Actor submits the queries previously composed to execute the search.</li> <li>3 The AXMEDIS Query Support for Clients, using the AXMEDIS Query Support, submits the Actor’s query to each of the chosen search “places” by using the corresponding specific interface: (i) Collector Engine Query Support Interface, (ii) AXEPTool Query Support Interface and (iii) AXMEDIS Database Manager (iv) AXMEDIS Data Model Query Support</li> <li>4 AXMEDIS Query Support merges the results all together and return the complete list to the AXMEDIS Query Support for Clients.</li> <li>5 AXMEDIS Query Support for Clients transcodes the result to the Actor in an adequate manner, i.e. in such a way that the Actor can understand: (i) from which source an object come (ii) which are the restriction on the object (iii) etc...</li> <li>6 The Actor can select a content. This selected content is requested from the Programme and Publication Tool Engine (content production on demand) by the distribution server. The distribution channel and client profiles are transmitted together with this request.</li> <li>7 The Programme and Publication Tool Engine initiates a work flow (WF) for the formatting of the content.</li> <li>8 After finishing the WF the object is sent to the distribution server, which forwards it to the requesting Actor.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Query Support for Clients (with reduced query functionality)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Web services should be adopted at the communication layer
<b>Issues</b>	The Formatting tool can get the objects from the database directly from the AXMEDIS database Manager and in particular from the AXMEDIS Object Loader and Saver.

### 9.2.3 Bookmark a query

<b>UCId</b>	UC9.2.3
<b>Use case</b>	Bookmark a query
<b>Description</b>	An Actor can save an executed query for future reuse.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	The bookmarked query must already be executed
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, after the execution of a query asks to store the query inside the query bookmarks that is in his/her personal profile</li> <li>2 The AXMEDIS Query Support User Interface save the query in the user profile allowing the user to choose a name and description for the bookmarked query. The query will be visible only to the user that have bookmarked it.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Query Support for Clients (with reduced query functionality)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 9.2.4 Retrieve a bookmarked query

<b>UCId</b>	UC9.2.4
-------------	---------

<b>Use case</b>	Retrieve a bookmarked query
<b>Description</b>	An Actor can retrieve a query stored in his bookmark collection for submitting the same query or create a new query starting from the bookmarked one.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	At least a query must exist in the bookmark of the user profile
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, recall from the bookmark in the user profile a stored query</li> <li>2 The AXMEDIS Query Support User Interface show the query in the query environment for issuing that query of for modifying it</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Query Support for Clients (with reduced query functionality)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 9.2.5 Organize bookmarked queries

<b>UCId</b>	UC9.2.5
<b>Use case</b>	Organize bookmarked queries
<b>Description</b>	An Actor can organize the queries in the bookmark dividing them in folders according her/his needs.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, recall from his/her user profile the bookmarks</li> <li>2 The Actor can create folder, rename folder, delete folder, insert query in a folder or removing queries from folders that belongs to his/her user profile</li> <li>3 The Actor confirms the new configuration of the bookmarks</li> <li>4 The AXMEDIS Query support save the new bookmark collection in the user profile</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	3a. The Actor cancel the modifications Query Support for Clients (with reduced query functionality)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	An user interface similar to that of Internet browser can be adopted.
<b>Issues</b>	None

### 9.2.6 Save an incomplete query

<b>UCId</b>	UC9.2.6
<b>Use case</b>	Save an incomplete query
<b>Description</b>	An Actor can save an incomplete or not executed query for future reuse.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, during the composition of a query asks to store the query inside the personal user profile</li> <li>2 The AXMEDIS Query Support User Interface saves the query in the user profile allowing the user to choose a name and description for the stored query.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Query Support for Clients (with reduced query functionality)

<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 9.2.7 Retrieve an incomplete query

<b>UCId</b>	UC9.2.7
<b>Use case</b>	Retrieve an incomplete query
<b>Description</b>	An Actor can retrieve a query stored in his/her user profile for submitting the same query or create a new query starting from the stored one.
<b>Actors</b>	Content Integrator, Content Distributor, Content Consumer
<b>Assumptions</b>	At least a query must exist in the user profile
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, recall from the user profile a stored query</li> <li>2 The AXMEDIS Query Support User Interface shows the query in the query environment for issuing that query of for modifying it</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Query Support for Clients (with reduced query functionality)
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 10 AXMEDIS AXEPTools for P2P distribution on B2B

### 10.1 AXEPTool for P2P on B2B

#### 10.1.1 Discovery and connection of peers on B2B P2P network

<b>UCId</b>	UC10.1.1
<b>Use case</b>	Discovery and connection of peers on B2B P2P network
<b>Description</b>	The user wants to discover and connect to one or more peers already connected on the P2P network.
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	The user launches the AXEPTool.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user press the “Connect” button in the GUI</li> <li>2 The AXEPTool starts a discovery protocol to find out on the network one or more participants in the AXMEDIS P2P network.</li> <li>3 The AXEPTool receives a list of hosts enabled to accept incoming connections.</li> <li>4 The AXEPTool tries to establish one or more connections. If a connection succeeds identities of local and remote host are exchanged in the handshaking (by means of digital signatures).</li> <li>5 Hosts without a certified identity cause the handshaking to fail. They ARE NOT allowed to join the AXMEDIS community.</li> <li>6 The remote host can refuse the connection because is busy. Then other remote hosts are contacted</li> <li>7 After a timeout, if no connection succeed the AXEPTool popup a “Unrecoverable Error” message to the user.</li> </ol>
<b>Post-conditions</b>	<p>If one or more connections succeed the user see the status “CONNECTED” in the status bar. Otherwise, a error message is popped-up</p> <p>When the connection is established the AXEPTool is ready to exchange messages with the other participants in the community.</p>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 10.1.2 Report P2P downloads/uploads network traffic

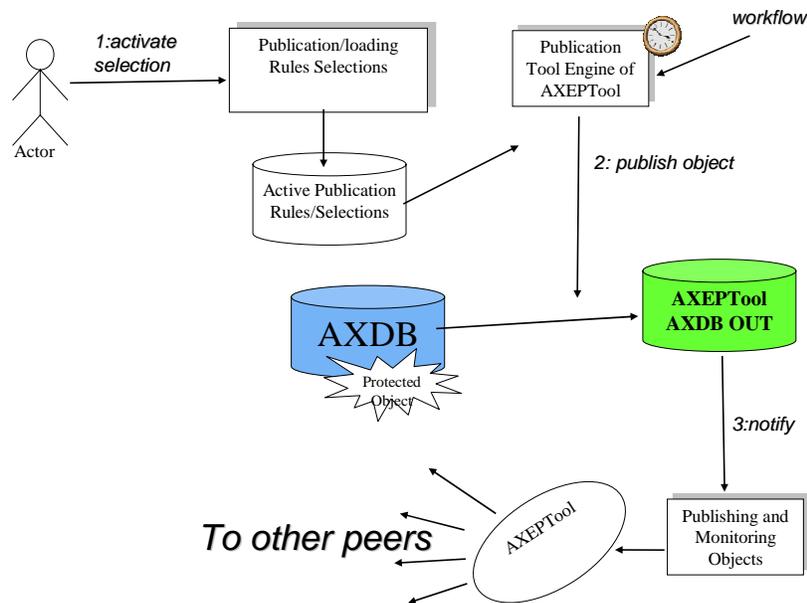
<b>UCId</b>	UC10.1.2
<b>Use case</b>	Report P2P downloads/uploads network traffic
<b>Description</b>	AXEPTool provides real-time, auto-refreshing, P2P network traffic reports for Downloads and Uploads. Moreover, uploads/downloads can be suspended, terminated, and resumed.
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	AXEPTool is connected to the network, some uploads/download are running .
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the “Download/Upload Table” in the UI</li> <li>2 Data regarding the messages exchanged are presented to the user</li> <li>3 The user selects one upload/download and suspend/resume/terminate the selected session</li> </ol>
<b>Post-conditions</b>	A download/upload is suspended/resumed/terminated
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 10.2 Publication and loading AXMEDIS Objects of AXEPTool

### 10.2.1 Creation of a publishing rule for the AXEPTool

<b>UCId</b>	UC10.2.1
<b>Use case</b>	Creation of a publishing rule for the AXEPTool
<b>Description</b>	The Publication Tool Engine is based on the AXCP and allows the user to execute publication rules in two ways: by scheduler and from the Rule Editor User Interface
<b>Actors</b>	Content Owner.
<b>Assumptions</b>	One or more objects are stored in the AXMEDIS Data Base
<b>Steps</b>	1a) The user opens the AXCP Editor 2a) The user fills the data required to build a new publication rule. These data include the query/selection to identify the objects to be published. Or alternatively 1b) The user manually selects one or more AXMEDIS objects in the AXMEDIS Data Base  in both cases  3. The Publication Tool Engine of AXEPTool saves the new rule in the AXEPTool Active Publication Rules/Selections for the execution based on the AXCP scheduler
<b>Post-conditions</b>	A set of AXMEDIS objects is available on the P2P network.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

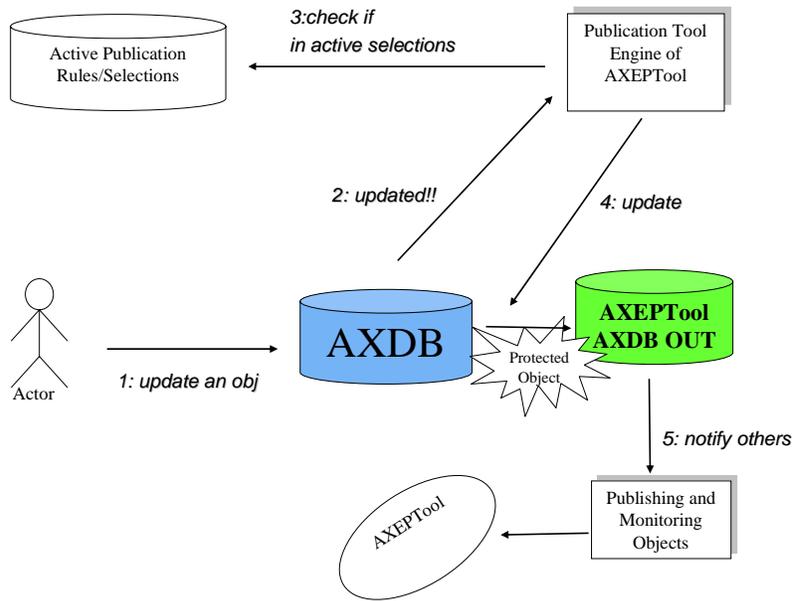
### 10.2.2 Automatic publication of a selection of objects on the AXEPTool



<b>UCId</b>	UC10.2.2
<b>Use case</b>	Automatic publication of a selection of objects on the AXEPTool.
<b>Description</b>	The Actor wants to publish one or more AXOB on the AXEPT Tool network.
<b>Actors</b>	Aggregator, Producers
<b>Assumptions</b>	The user creates a Selection of one or more AXOB. The AXMEDIS Objects are stored in the AXDB.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor, through the AXCP Editor and the Selection editor create a rule and a selection as input parameter for the AXEPTool Active Publication Active Rules/Selections <ul style="list-style-type: none"> <li>o the Selection becomes active</li> <li>o the Actor is allowed to modify the default activation period in the scheduler</li> </ul> </li> <li>2 According to activation periods the AXCP Scheduler publishes each objects of the Selection on the AXEPTool OUT AXDB.</li> <li>3 The AXEPTool OUT AXDB advises the Publishing and Monitoring Objects which provides to broadcast the event to all its counterparts on the network</li> </ol>
<b>Post-conditions</b>	One or more AXOB are stored in the AXEPTool OUT AXDB. The other Peers are notified that one or more AXOB have been published by another Peer.
<b>Variations</b>	Publication of AXMEDIS objects on AXEPTool can be also made using the AXMEDIS Workflow Manager.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXEPTool IN/OUT AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	Every time that an object is published its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.

### 10.2.3 Automatic updating of a modified object on the AXEPTool



<b>UCId</b>	UC10.2.3
<b>Use case</b>	Automatic updating of a modified object on the AXEPTool.
<b>Description</b>	The user decides to activate the procedure of updating. The aim is to update an AXOB already published.
<b>Actors</b>	Aggregator, Producers
<b>Assumptions</b>	The AXOB is already been published. The AXOB has been modified in the AXDB by the owner.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user activates the updating process.</li> <li>2 The local AXDB advises the Publication Tool Engine of AXEPTool that the object has been updated.</li> <li>3 The AXOB has to belong to one of the active Selections or list of objects.</li> <li>4 Publication Tool Engine of AXEPTool updates the AXOB contained into the AXEPTool OUT AXDB.</li> <li>5 The other Peers in the network are notified.</li> </ol>
<b>Post-conditions</b>	The AXOB is stored in the AXEPTool OUT AXDB in the new version. The other Peers are notified that one or more AXOB have been updated by another Peer.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXEPTool IN/OUT AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	Every time that an object is published its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.

### 10.2.4 Manual Publication of AXMEDIS Objects with the AXEPTool

<b>UCId</b>	UC10.2.4
<b>Use case</b>	Manual Publication of AXMEDIS Objects with the AXEPTool
<b>Description</b>	The user wants to publish an AXMEDIS object in the AXEPTool. Objects are copied into the Output Database of the AXEPTool.
<b>Actors</b>	Content Owner, distributor, integrator
<b>Assumptions</b>	One or more objects are presently accessible in the AXMEDIS Data Base or are in the file system
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user invokes the AXEPTool</li> <li>2 The user selects objects in the AXMEDIS Data Base.</li> <li>3 The user start the manual publishing procedure</li> <li>4 If a selected object cannot be published the user is informed and the procedure aborted</li> <li>5 If all the object(s) can be published, the object or the objects are copied in the AXEPTool OUT AXMEDIS Output Database</li> </ol>
<b>Post-conditions</b>	A selection of AXMEDIS object is available on the P2P network.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	Every time that an object is published its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.

### 10.2.5 Producing a query to search on the AXEPTool network

<b>UCId</b>	UC10.2.5
<b>Use case</b>	Producing a query to search on the AXEPTool network
<b>Description</b>	The user wants to produce a query in order to search AXMEDIS objects on P2P network.
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	The peer is connected to the P2P network.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the Advanced Query UI to produce a technical query.</li> <li>2 The user fills in the fields with the necessary information.</li> <li>3 The user starts the query.</li> </ol>
<b>Post-conditions</b>	A query result sheet is created and added to the AXEPTool UI A query message is produced and sent to the network. Results are collected and presented in the query result sheet in the UI.
<b>Variations</b>	None

<b>Asynchronous actions</b>	The user can launch more than one query at a time.
<b>Design suggestions</b>	The fields in the query can be as complex as the metadata model used to describe AXMEDIS Objects. Thus depending on metadata, the GUI can change the fields presented to the user. This can be unified with the Query Support
<b>Issues</b>	Manual browsing should be available as well.

### 10.2.6 View/Manage query results coming from the AXEPTool

<b>UCId</b>	UC10.2.6
<b>Use case</b>	View/Manage query results coming from the AXEPTool
<b>Description</b>	The user wants to manage with the results of the querying process.
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	A query has been sent.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the query result sheet for the query he/she is interested.</li> <li>2 Results received (from others peers) are presented in form of query-hits (the datum containing all the information related to a remote AXMEDIS Object) relevant to the query.</li> <li>3 The user can sort/delete/make selections on the query result sheet.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	This can be unified with the Query Support. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	Every time that a descriptor of an object received to be presented to the user its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.

### 10.2.7 Active query pool management for the AXEPTool

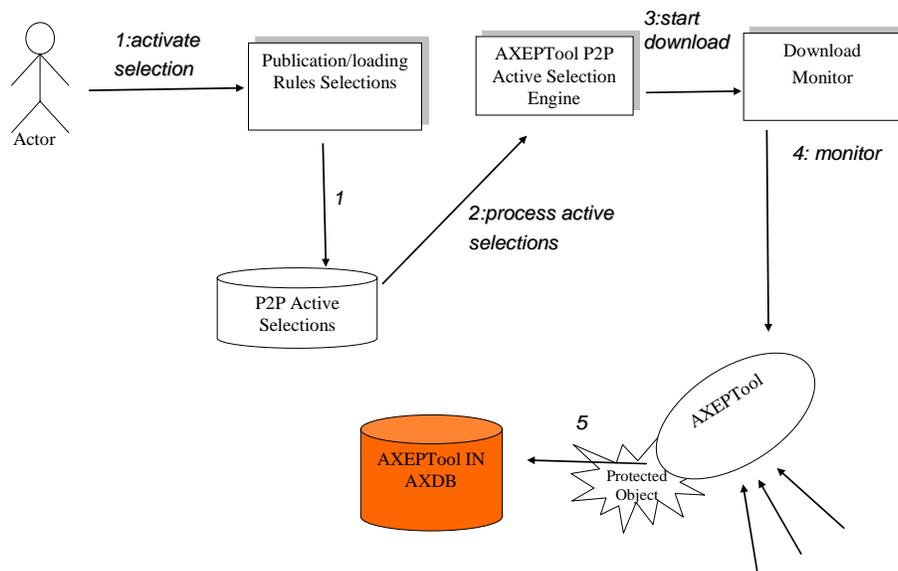
<b>UCId</b>	UC10.2.7
<b>Use case</b>	Active query pool management for the AXEPTool
<b>Description</b>	The user wants to keep the AXEPTool up-to-date with respect to a particular query.
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	One or more queries have been sent to the AXEPTool (that keep them inside) for downloading for automating the download from the network into the IN DB. Their query result can be requested on the GUI of the AXEPTool (or of the Publication/Loading Rules/Selections Editor) selecting the query from the list of active queries.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user selects one query for downloading</li> <li>2 The user may select a time-interval for the query to be re-sent to the network</li> <li>3 The user can cancel a query</li> <li>4 The user can put the query in sleep</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	This can be unified with the Query Support. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	None

### 10.2.8 Downloading an AXMEDIS object

<b>UCId</b>	UC10.2.8
<b>Use case</b>	Download AXMEDIS Object
<b>Description</b>	The user chooses to download a selection of AXMEDIS objects available on the P2P network.
<b>Actors</b>	The AXEPTool user.

<b>Assumptions</b>	One or more objects are shown as available in the P2P network within a query result sheet.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor selects one or more objects in a query result sheet and starts the download .</li> <li>2 AXEPTool verifies DRM rules, protections and licensing aspects.</li> <li>3 A download session is started. A download session sheet is created and inserted in the GUI.</li> <li>4 The lists of selected objects can be transformed in an active query to keep them updated into the IN DB.</li> </ol>
<b>Post-conditions</b>	Once a download session successfully terminates, the downloaded object is stored in the AXEPTool IN AXDB. Every time that an object is downloaded its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.
<b>Variations</b>	None
<b>Asynchronous actions</b>	The Actor can start, suspend, cancel or resume the download session of an object
<b>Design suggestions</b>	Feedback on download status must be implemented. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
<b>Issues</b>	none

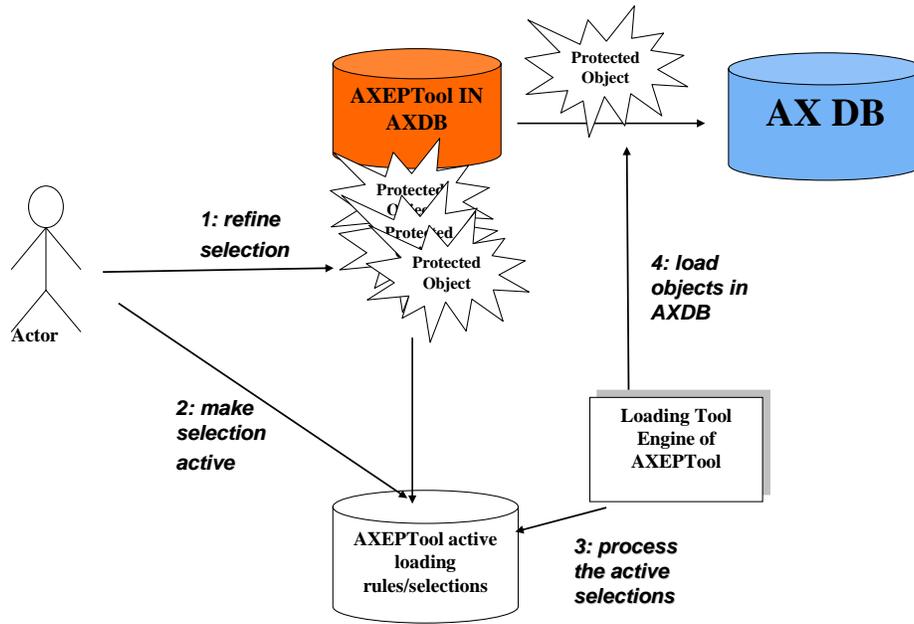
### 10.2.9 Automatic downloading of a selection of objects available in the P2P network



<b>UCId</b>	UC10.2.10
<b>Use case</b>	Automatic downloading of a selection of objects available in the P2P network.
<b>Description</b>	The Actor wants to load within the local AXMEDIS Database one or more AXOB which belongs to a given Selection of AXMEDIS objects available on the AXEPTool network.
<b>Actors</b>	Aggregator, Content Provider, Publisher
<b>Assumptions</b>	The Actor has previously created a Selection of one or more AXOB on the AXEPTool network which satisfy some Actor's needs by using the AXQS User Interface integrated within the Publication/Loading Rules/Selections Editor.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The User activates the Selection by using the Publication/Loading Rules/Selections Editor.</li> <li>2 AXEPTool P2P Active Selection Engine elaborates the active Selections contained in the P2P Active Selections.</li> <li>3 AXEPTool P2P Active Selection Engine downloads each AXOB of the Selection from the network.</li> <li>4 The AXEPTool Monitor has the duty of monitoring the object. Every time that an object is downloaded its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.</li> <li>5 The object is stored in the AXEPTool IN AXDB.</li> </ol>
<b>Post-conditions</b>	The Actor can play, run, visualize, etc, each object accordingly to the related DRM rules.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

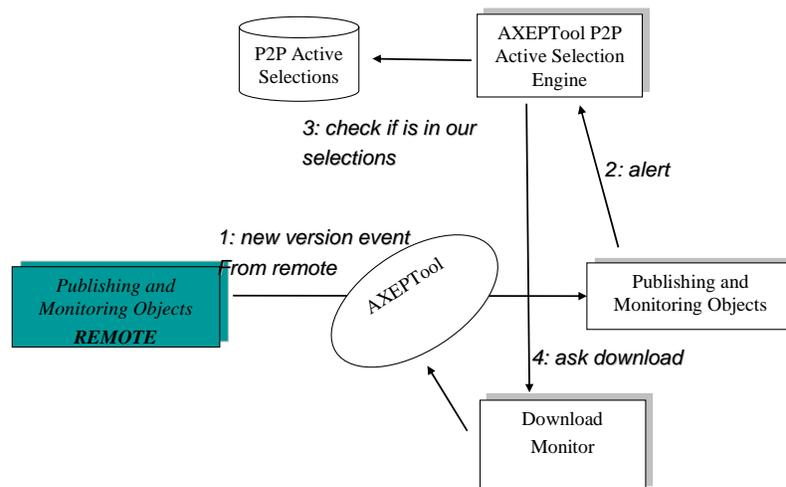
**10.2.10 Selecting objects for the AXDB from the those downloaded**



<b>UCId</b>	UC10.2.10
<b>Use case</b>	Selecting objects for the AXDB from the those downloaded
<b>Description</b>	The Actor decides which AXOB downloaded is interested in.
<b>Actors</b>	Aggregator, Content Provider, Publisher
<b>Assumptions</b>	The Actor has tried the loaded objects, according to the related DRM rules.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 the Actor selects only the AXOB he/she is interested in. This is what can be called and Active Selection for loading.</li> <li>2 the chosen objects from those that have been downloaded have to be moved from the IN AXDB to the AXDB. To this end, an AXCP rule is activated with a specific rule for metadata mapping.</li> <li>3 Each activated rule for each object or set of objects can be periodically activated to update the objects into the AXDB on the basis of changes performed on objects into the IN AXDB that come from other AXMEDIS factories.</li> <li>4 The AXCP Engine and Scheduler elaborates the rules for loading on the chosen objects</li> <li>5 Each related object is loaded into the AXDB.</li> </ol>
<b>Post-conditions</b>	The AXOB are in the AXDB.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 10.2.11 Automatic loading new versions of AXMEDIS Objects for the AXEPTool



<b>UCId</b>	UC10.2.11
<b>Use case</b>	Automatic loading new versions of AXMEDIS Objects for the AXEPTool
<b>Description</b>	Thee peer is informed of updating in published objects. The scenario is performed in automatic way, by interoperability of Publishing and Monitoring Objects modules of remote AXEPTools.
<b>Actors</b>	AXEPTool via AXCP
<b>Assumptions</b>	One or more Active Selections for loading have already been produced. A new version of a previously downloaded object is published.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Publication and Monitoring Objects is informed of the updating.</li> <li>2 An AXCP engine for loading alias AXEPTool P2P Active Selection Engine is alerted by Publication and Monitoring Objects.</li> <li>3 An AXCP engine for loading alias AXEPTool P2P Active Selection Engine verifies if the updated object belongs to Active Selections.</li> <li>4 AXEPTool downloads the new version of the object and if its eligible as a « loadable » object it is loaded in the AXDB from the IN AXDB.</li> </ol>
<b>Post-conditions</b>	Every time that an object is downloaded its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 10.2.12 Automatic loading new AXMEDIS Objects with the AXEPTool

<b>UCId</b>	UC10.2.12
<b>Use case</b>	Automatic loading new AXMEDIS Objects with the AXEPTool
<b>Description</b>	Thee peer is informed of publishing of new objects. The scenario is performed in automatic way, by interoperability of Publishing and Monitoring Objects modules of remote AXEPTools.
<b>Actors</b>	AXEPTool via AXCP
<b>Assumptions</b>	One or more Active Selections have already been performed.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Publication and Monitoring Objects is informed of the new publication.</li> <li>2 An AXCP engine alias AXEPTool P2P Active Selection Engine is alerted by Publication and Monitoring Objects.</li> <li>3 An AXCP engine alias AXEPTool P2P Active Selection Engine verifies if the new published objects matches certain features in the Active Selections.</li> <li>4 An AXCP engine alias AXEPTool P2P Active Selection Engine loads the new object.</li> <li>5 Eventually metadata mapping is performed in the rule script</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Every time that an object is downloaded or loaded its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.
<b>Issues</b>	None

### 10.2.13 Manual Loading of AXMEDIS Objects with the AXEPTool

<b>UCId</b>	UC10.2.13
<b>Use case</b>	Manual Loading of AXMEDIS Objects with the AXEPTool
<b>Description</b>	The user wants to move an AXMEDIS object from the AXEPTool IN AXMEDIS Data Base to the AXMEDIS Data Base.
<b>Actors</b>	Content Provider, Aggregator, Integrator
<b>Assumptions</b>	One or more objects are stored in the AXEPTool IN AXMEDIS Data Base

Steps	<ol style="list-style-type: none"> <li>1 The user, is invoked by the Publication/Loading Rules/Selection Editor, select one or more objects in the AXEPTool IN AXMEDIS Data Base.</li> <li>2 The user activate the loading</li> <li>3 The AXCP engine for Loading activate rules to move selected objects from the AXEPTool IN AXMEDIS Data Base to the AXMEDIS Data Base on the basis of the right metadata mapping: <ol style="list-style-type: none"> <li>3.1 The objects are selected by the provided Selections</li> <li>3.2 The objects are moved</li> </ol> </li> </ol>
Post-conditions	A selection of AXMEDIS object is available in the AXEPTool.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	Every time that an object is downloaded or loaded its integrity has to be verified in order to allow publishing its metadata and certifying that they are coherent with the description of the object.

#### 10.2.14 Creation of a loading rule for the AXEPTool

UCId	UC10.2.14
Use case	Creation of a loading rule for the AXEPTool
Description	The Loading Tool Engine (alias AXCP Engine for loading) allows the user to build loading rules in two way: by example and by the Publication/Loading Rules/Selection Editor.
Actors	Content Provider, Aggregator, Integrator
Assumptions	One or more objects are stored in the AXEPTool IN AXMEDIS Data Base
Steps	<ol style="list-style-type: none"> <li>1a) The user opens the Publication/Loading Rules/Selection Editor of the Loading tool engine</li> <li>2a) The user fills the data required to build a new loading rule: mainly period of activity, metadata mapping disable or enable, list of objects to be manipulated by the rule.</li> <li>1b) The user may manually selects an AXMEDIS objects for composing the call selecting them from the IN AXDB, query support can be used for this</li> </ol> <p>in both cases The Loading Tool Engine saves the new rule in the Loading Rules repository.</p>
Post-conditions	A selection of AXMEDIS object is available on the AXEPTool.
Variations	None
Asynchronous actions	None
Design suggestions	Query support and AXMEDIS database manager can be used for creating the support for queries and the AXEPTool IN AXMEDIS Data Base
Issues	None

#### 10.2.15 Preview an AXMEDIS object content coming from AXEPTool

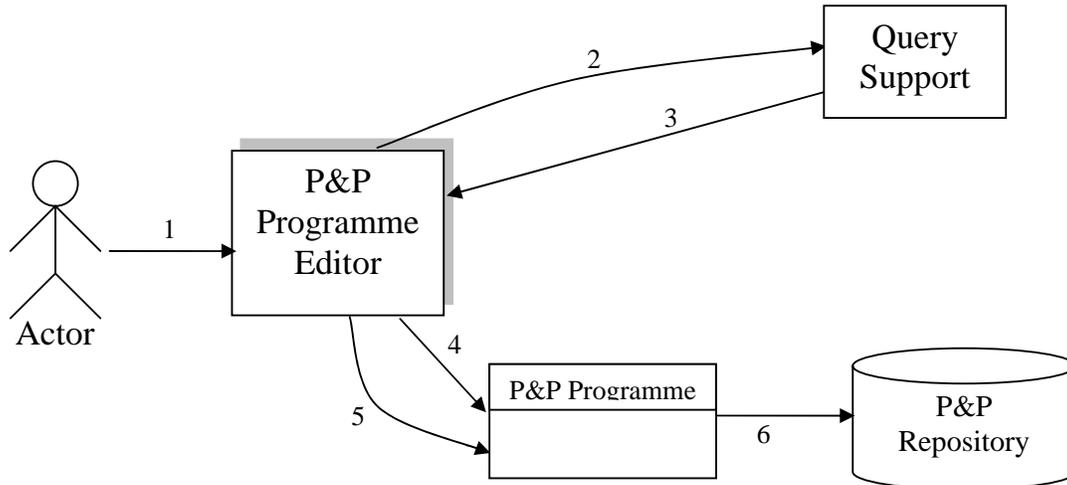
UCId	UC10.2.15
Use case	Preview an AXMEDIS object content coming from AXEPTool
Description	According to the object media type, the AXEPTool provides a preview modality. The object preview should be performed by an associated software for every media type.
Actors	The AXEPTool user.
Assumptions	The object is in the AXEPTool IN AXMEDIS Data Base.
Steps	<ol style="list-style-type: none"> <li>1 The user chooses to preview an object.</li> <li>2 The AXEPTool uses a suitable player for this task..</li> <li>3 The object is previewed or an error message should be prompted if not possible.</li> </ol>
Post-conditions	None

<b>Variations</b>	In the case the user (for instance and editor or a producer) wants to perform operations on the preview to evaluate the usability of the content, the AXEPTool according to the specific license allows to edit the preview version of the object.
<b>Asynchronous actions</b>	The user can cancel, stop, close or replay the preview.
<b>Design suggestions</b>	The preview is performed with the players available.
<b>Issues</b>	None

## 11 Programme and Publication Engine Tools

### 11.1 Programme and Publication Programme Production

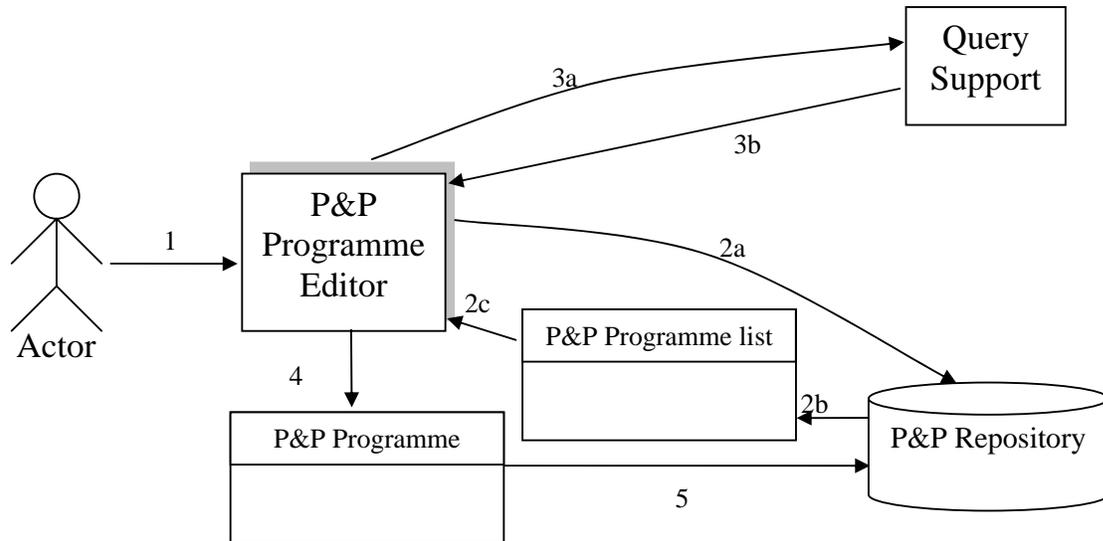
This section describes how the Programme and Publication Programmes are produced.



<b>UCId</b>	UC10.1
<b>Use case</b>	Programme and Publication Programme Production
<b>Description</b>	To create/define/edit a programme for certain channel
<b>Actors</b>	A programme manager
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Actor initiates GUI in the Programme and Publication Editor</li> <li>2 The Actor submits queries to Query Support for a list of AXMEDIS objects</li> <li>3 Query Support returns a list of AXOIDs (see UCs in section 3.1.2)</li> <li>4 The Programme and Publication Editor GUI allows the actor to select AXOIDs</li> <li>5 The Actor specifies what (AXOID), where (channel), when (schedule), etc..</li> <li>6 The Programme is saved in the Programme and Publication Repository which can be re-used</li> </ol>
<b>Post-conditions</b>	By default the Programme is “inactive” at the end of the programme production, until the Actor activated/published it.
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Requires connection/interface to various other modules including the AXMEDIS Query Support,, etc.
<b>Issues</b>	None

### 11.2 Programme and Publication Programme Editing

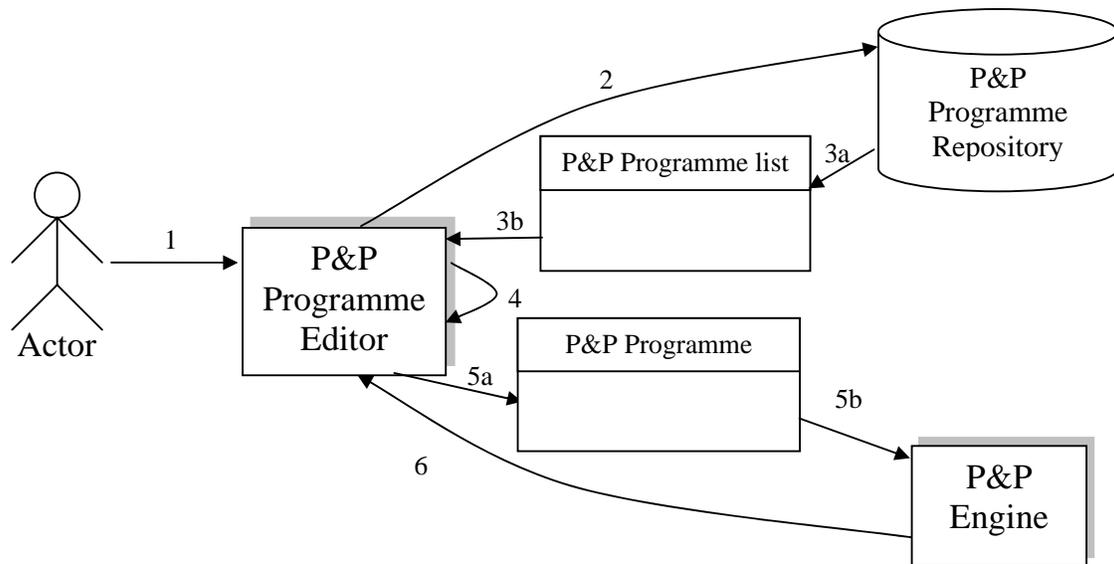
This section describes how the Programme and Publication Programme are manipulated.



<b>UCId</b>	UC10.2
<b>Use case</b>	Programme and Publication Programme Editing
<b>Description</b>	Using the Programme and Publication Editor to edit an existing P&P Programme
<b>Actors</b>	A programme manager
<b>Assumptions</b>	There are one or more predefined/created programme which can be reused, in the P&P Repository
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Actor initiates the P&amp;P GUI</li> <li>2 (2a) the user browse the existing programmes in the P&amp;P Repository and (2b) selects P&amp;P Programme for a Programme list for (2c) load the selected P&amp;P Programme</li> <li>3 a. Actor queries for additional objects using Query Support b. Query Support returns a list of AXOID</li> <li>4 Actor edits the Programme (e.g. time, channel)</li> <li>5 The revised P&amp;P Programme is saved to the Programme and Publication Repository as a new Programme or overwriting the original Programme</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Requires connection/interface to various other modules including Query Support User Interface, AXMEDIS Plugin Manager, AXMEDIS Workflow Plugin
<b>Issues</b>	Getting a 'selection' from the Query Support web interface is an issue at the moment. The Query Support web interface has to provide return string for "selection" in the same manner as the selected AXOID.

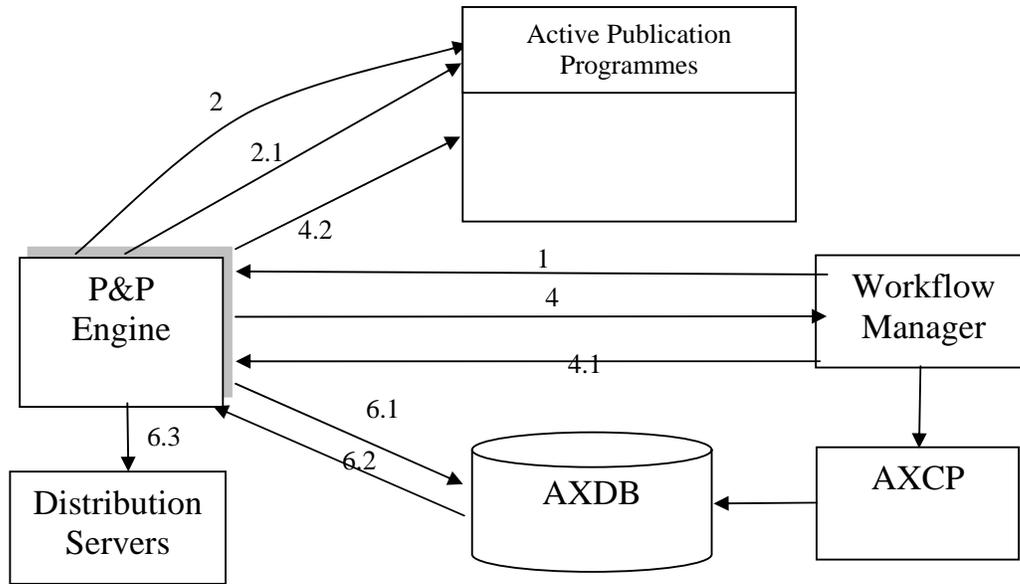
### 11.3 Activation of Programme and Publication Programmes

This section describes how the Programme and Publication Programmes are activated. For the publication of AXMEDIS Objects



<b>UCId</b>	UC10.3
<b>Use case</b>	Programme Publication
<b>Description</b>	The user decide the publish (“activate”) the programme
<b>Actors</b>	A programme manager
<b>Assumptions</b>	A completed programme
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The actor uses the Programme and Publication Editor GUI</li> <li>2 If the programme has not been loaded, the user can select and load the programme, for final checking</li> <li>3 The programme/schedule is returned from the repository</li> <li>4 A GUI to allow the user to activate/publish the programme (4a) to the P&amp;P Engine (4b)</li> <li>5 The P&amp;P Editor checks and reports an invalid programme</li> <li>6 A confirmation on the success of the publication</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Request for activation by WF
<b>Asynchronous actions</b>	The user can modify/cancel this action before the schedule distribution. Note that the schedule distribution time is not the same as the programme schedule time. Schedule distribution time is before the actual programme time, taking into account the time required for distribution and/or any formatting requirement
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 11.4 Launch of Programme and Publication Programmes from Workflow



<b>UCId</b>	UC10.4
<b>Use case</b>	Launch of Programme and Publication Programme from Workflow
<b>Description</b>	This is an active engine which monitors the system clock to ensure that one or more scheduled and published programmes is delivered in time for the actual consumption.
<b>Actors</b>	Workflow User
<b>Assumptions</b>	The engine is running with correct system clock. Distribution channel profile including the bandwidth – with an estimated time for the actual delivery and time required for formatting (if On-Demand is needed).
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Workflow sends a P&amp;P programme to be activated</li> <li>2 The Engine checks if the P&amp;P Programme is already running                         <ol style="list-style-type: none"> <li>2.1 Engine searches active programmes</li> <li>2.2 The matching programme is removed if found</li> </ol> </li> <li>3 check source and target format</li> <li>4 if profiles mismatch                         <ol style="list-style-type: none"> <li>4.1 send request to AXCP via Workflow</li> <li>4.2 receive new object ID from workflow</li> <li>4.3 update the programme with new object ID</li> </ol> </li> <li>5 calculate distribution time based on distribution schedule and distribution server profile</li> <li>6 programme is scheduled for distribution</li> <li>7 at their scheduled distribution time                         <ol style="list-style-type: none"> <li>7.1 request Objects from AXDB</li> <li>7.2 Receive objects (from AXDB according to Scenarios v3.6)</li> <li>7.3 Send to Distribution Servers</li> </ol> </li> </ol>
<b>Post-conditions</b>	Update the P&P Active Programme Repository together with the status of the process

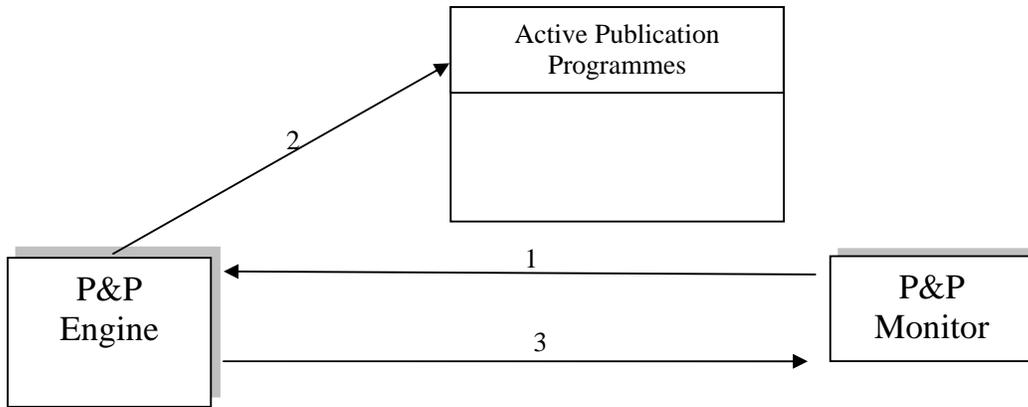
<b>Variations</b>	Content production On-Demand (request from the On-Demand section via Workflow). In this case the content is produced only for the requesting distribution server/terminal and to be processed and distributed immediately. In addition to “Activate” request, Workflow can send: 1 a “Kill” request. In this case, the Engine will remove the scheduled Programme 2 a “List” request. In this case, the Engine will return a list of scheduled programmes 3 a “Status” request. In this case, the Engine will return the status of the relevant scheduled programme.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Requires connection/interface to various other modules including AXCP, Distribution servers, Workflow, AXMEDIS Plug-in Manager, AXMEDIS Configuration Manager
<b>Issues</b>	Pending research on to approach to estimate time requirements for AXCP and distribution server profile in order to optimise scheduling of distribution

## 11.5 Trial Pre-activation of Programme and Publication Programme

This section describes how the Programmes for Programme and Publication are pre-activated to simulate, test and be prepared.

<b>UCId</b>	UC10.5
<b>Use case</b>	Programme Publication Pre-activation
<b>Description</b>	The user decide the publish (“quick trial”) or (“full trial”) the programme
<b>Actors</b>	A programme manager
<b>Assumptions</b>	A completed programme
<b>Steps</b>	1 Steps 1-4 the same as UC10.3 (Programme Publication) 2 The P&P Editor GUI to allow the user to activate/publish the Programme as a trial (quick trial or full trial) 3 A confirmation on the success of the trial-run when completed
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	The user can modify/cancel this action before the completion of the trial process.
<b>Design suggestions</b>	Use of variable to signify the level of test (0 for activation, 1 for quick test, 2 for full test). All connected modules should understand this variable (e.g. AXCP, P&P Engine).
<b>Issues</b>	1. A quick trial would complete each stage for publication without requiring the engines such as the formatting engine to actually format the object but simply acknowledge if it can format the object from the source object to a given target representation. 2. The full trial completes a publication without final distribution (optional)

## 11.6 Monitoring of Programme and Publication Engine



<b>UCId</b>	UC10.6
<b>Use case</b>	Monitoring of Programme and Publication Engine
<b>Description</b>	This use case begins when a programme manager wishes to monitor programmes running on the P&P Engine. The monitor application is launched and the actor can view and manage active P&P programmes.
<b>Actors</b>	Programme Manager
<b>Assumptions</b>	A P&P Engine is running at a known location.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor starts the P&amp;P Engine Monitor and connects to a P&amp;P Engine</li> <li>2 The Monitor Requests a list of active programmes</li> <li>3 The Engine returns the current list</li> <li>4 The Monitor updates the GUI window</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	The Actor can select an active P&P programme from the list and “Kill” it The P&P Engine notifies the Workflow that the specific P&P programme has been killed
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	none
<b>Issues</b>	none

## 12 AXMEDIS AXEPTOOLS for Satellite Data Broadcast on B2B

### 12.1 AXMEDIS B2B Client Application

#### 12.1.1 B2B Client Installation

<b>UCId</b>	UC12.1.1
<b>Use case</b>	B2B Client Installation
<b>Description</b>	A professional user installs the B2B Client Application (hardware and software) on the Computer of either an AXMEDIS Distributor or an AXMEDIS Receiving Station (controlled by an AXMEDIS Distributor)
<b>Actors</b>	The AXMEDIS professional user
<b>Assumptions</b>	The professional PC is connected to a satellite dish, correctly pointed to the satellite providing the Data Broadcast. The professional PC has a PCI slot, an Ethernet port, or an USB connector free for installing the DVB-IP adapter. The user PC has a working connection to the Internet.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The professional user obtains a DVB-IP satellite adapter suitable for the professional PC configuration (depending on operating system, available ports, etc.) and fully supported by the AXMEDIS B2B Client Application</li> <li>2 The professional user physically installs the DVB-IP adapter according to the installation instructions provided by the manufacturer</li> <li>3 The professional user connects the DVB-IP adapter to the satellite dish</li> <li>4 The professional user boots the PC and installs any required software driver or application, as specified by the manufacturer in the installation instructions, and in the AXMEDIS Client Application user manual</li> <li>5 The professional user configures the DVB-IP adapter according to instructions</li> <li>6 The professional user checks that the satellite signal is received correctly</li> <li>7 The professional user has a special Setup to install the AXMEDIS B2B Client Application</li> <li>8 The professional user runs the AXMEDIS B2B Client Application Setup</li> <li>9 The professional user follows the steps of the installation</li> </ol>
<b>Post-conditions</b>	The professional user installs other needed Applications useful to treat associated actions with certain AXMEDIS Object.
<b>Variations</b>	The whole B2B Client (hardware and software) could be integrated in a unique box. The AXMEDIS Box could be simply installed in a professional environment (e.g., a server farm) and integrated in a rack.
<b>Asynchronous actions</b>	Interactions with operating system components (e.g., firewall) or installed software (e.g., antivirus) could stop the DVB-IP adapter from working correctly. Repeated installation of drivers, or installation of out-of-date drivers, or installation procedure not compliant with instructions, might stop the DVB-IP adapter from working correctly.
<b>Design suggestions</b>	A list of compatible adapters should be prepared. Full installation instructions should be given to the user.
<b>Issues</b>	If the satellite signal is not received correctly, there could be a problem in the pointing of the satellite dish, or in the satellite cable, or in the DVB-IP installation. Problems must be solved before proceeding. Occasional loss of signal (e.g., in presence of heavy rain or wind) does not represent a major problem; however, it may impact the fruition of service during and after the problem. It is recommended that the satellite dish installation be done by a professional.

#### 12.1.2 B2B Client Customization

<b>UCId</b>	UC12.1.2
<b>Use case</b>	B2B Client Customization

<b>Description</b>	The user installs the AXMEDIS Client Application
<b>Actors</b>	The AXMEDIS professional user
<b>Assumptions</b>	The B2B Client Installation has been done successfully.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The professional user configures local and external firewall</li> <li>2. The professional user checks that previously installed software does not interfere with the correct functioning of the AXMEDIS B2B Client components (hardware and software)</li> <li>3. The professional user modifies (if necessary) some configuration files</li> <li>4. The professional user disables (if necessary) some complementary module depending on the local configuration (e.g., operating system)</li> <li>5. The professional user has a stable contact with the Satellite Data Broadcast Provider technical team</li> <li>6. The professional user keeps up to date the professional computer hosting the B2B receiving station at different layers (drivers, antivirus, service packs, additional modules)</li> <li>7. The professional user keeps up to date the B2B Client Application Component at different layers (drivers, software setup, additional modules)</li> <li>8. The professional user installs (if it is not already present) a software in order to remotely control the B2B receiving station in case of problems</li> </ol>
<b>Post-conditions</b>	The professional user will put a special label of quality in the B2B receiving station, certifying the state of art of his installation
<b>Variations</b>	Some scripts could check the correct status of the B2B receiving station and send to a central server detected anomalies.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

### 12.1.3 B2B Client Registration

<b>UCId</b>	UC12.1.3
<b>Use case</b>	B2B Client Registration
<b>Description</b>	The professional installer registers the B2B Client Application in order to access the AXMEDIS B2B service
<b>Actors</b>	The AXMEDIS professional user
<b>Assumptions</b>	The professional user has successfully installed the hardware and software AXMEDIS components.
<b>Steps</b>	<ol style="list-style-type: none"> <li>9. The professional user runs the AXMEDIS Client Application registration procedure</li> <li>10. The AXMEDIS B2B Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)</li> </ol>
<b>Post-conditions</b>	The B2B receiving station is ready to use the AXMEDIS B2B service and receive the AXMEDIS B2B Object.
<b>Variations</b>	The procedure to update the B2B receiving station profile could be automatic and hidden for the system.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	The Server shall manage the B2B receiving station profiles useful to address the content to a part of the B2B users.
<b>Issues</b>	None.

## 12.2 Enabling a B2B receiving station

<b>UCId</b>	UC12.2
<b>Use case</b>	Enabling a B2B receiving station

<b>Description</b>	The AXMEDIS Distributor registers a receiving station/device (controlled by him) in order to enable the station receiving AXMEDIS Content from the AXMEDIS B2B Carousel. The AXMEDIS B2B Client receives automatically the content by push.
<b>Actors</b>	The AXMEDIS Distributor
<b>Assumptions</b>	The AXMEDIS Distributor knows exactly all needed information for registering an authorized B2B station.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Distributors accesses to the AXMEDIS User Admin Interface (AXUAI).</li> <li>2. The AXMEDIS Distributor can manage (add/modify/delete) all receiving station controlled by him</li> </ol>
<b>Post-conditions</b>	The AXMEDIS B2B Client, which was just enabled to receive the AXMEDIS B2B carousel, will receive all needed notifications.
<b>Variations</b>	The AXMEDIS Distributor can create one or more groups, and then can associate a receiving station to one or more groups.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	Design a solid environment where the AXMEDIS B2B Client can be simply auto-updated.
<b>Issues</b>	None.

### 12.3 Downloading AXMEDIS Objects from AXEPTool by using Satellite Data Broadcast on B2B

<b>UCId</b>	UC12.3
<b>Use case</b>	Download AXMEDIS Object from AXEPTool by using Satellite
<b>Description</b>	The user chooses to download a selection of AXMEDIS objects available on the P2P network and to push this content to his authorized B2B receiving stations by using Satellite Data Broadcast on B2B
<b>Actors</b>	The AXEPTool user.
<b>Assumptions</b>	One or more objects are shown as available in the P2P network within a query result list.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Actor selects one or more objects</li> <li>2. The Actor chooses the Download Transfer mode (P2P, Satellite Data Broadcast)</li> <li>3. The Actor (after choosing Satellite Data Broadcast) selects one or more B2B receiving stations (controlled by him) for receiving the previously selected Object</li> <li>4. The Actor starts the download task in AXEPTool</li> <li>5. Verification of DRM rules, protections and licensing aspects</li> <li>6. Downloads status are showed in a particular view of the AXEPTool. The AXEPTool obtains it from the Push Server, by calling a specified API.</li> </ol>
<b>Post-conditions</b>	The downloaded object is stored in the Satellite Data Broadcast storage server before sending it.
<b>Variations</b>	None
<b>Asynchronous actions</b>	The Actor can start, suspend, cancel or resume the download task of an object
<b>Design suggestions</b>	Feedback on download status must be implemented.
<b>Issues</b>	None

#### 12.3.1 Pushing an AXMEDIS Object by B2B Carousel

<b>UCId</b>	UC12.3.1
<b>Use case</b>	Pushing AXMEDIS Content by B2B Carousel
<b>Description</b>	The distributor schedules the AXMEDIS Objects, received by the AXEPTool P2P network, for pushing those to the B2B authorized receiving stations. The AXMEDIS Content reaches multiple B2B sites simultaneously.

<b>Actors</b>	The AXMEDIS Distributor.
<b>Assumptions</b>	The AXMEDIS Distributor is authorized to use the Satellite Data Broadcast like delivery means.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Actor packages the downloaded content to be compatible with the Satellite Data Broadcast system</li> <li>2. The Actor selects the group of authorized receiving B2B stations to associate with the AXMEDIS Content</li> <li>3. The Actor associates the selected Object to a given Programme (the programme is charged of transmitting the Carousel sequence)</li> <li>4. The Actor schedules the Programme for transmission</li> </ol>
<b>Post-conditions</b>	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	The Actor can start, suspend, cancel or resume the Programme transmission of the Carousel.
<b>Design suggestions</b>	None.
<b>Issues</b>	None

#### 12.4 Automatic Content Reception via Satellite

<b>UCId</b>	UC12.4
<b>Use case</b>	Automatic Content Reception via Satellite
<b>Description</b>	The AXMEDIS B2B Client Application has detected an AXMEDIS Object addressed to him. He receives automatically the content by push.
<b>Actors</b>	The AXMEDIS B2B Client Application
<b>Assumptions</b>	The AXMEDIS B2B Client Application runs permanently on the AXMEDIS Distributor remote station like a daemon.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS B2B Client detects an AXMEDIS Object in the Electronic Programme Guide of the Satellite Data Broadcast</li> <li>2. The AXMEDIS B2B Client checks if it has all rights to listen the incoming transmission of the Object</li> <li>3. The AXMEDIS B2B Client launches all needed operations in order to receive the AXMEDIS Content</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	Design a solid environment where the AXMEDIS B2B Client can be simply auto-updated. The AXMEDIS Action Manager it is capable to do different actions on the basis of the different type of object received.
<b>Issues</b>	None.

#### 12.5 Content Delivery via Satellite

<b>UCId</b>	UC12.5
<b>Use case</b>	Content Delivery via Satellite
<b>Description</b>	The AXMEDIS B2B Client Application has successfully received an AXMEDIS Object addressed to him. He runs, either directly or by calling other applications, all actions associated with the Object. All actions should be executed at the end of the reception.
<b>Actors</b>	The AXMEDIS B2B Client Application and other Applications charged of applying actions on the AXMEDIS Object.
<b>Assumptions</b>	None.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Client Application receives the last bit of the current transmission and completes the AXMEDIS Object</li> <li>2. The AXMEDIS Client Application checks the correctness of the received Object (checksum, version numbering)</li> <li>3. The AXMEDIS Client Application loads the actions to be executed on the Object after reception</li> <li>4. The AXMEDIS Client Application parses the action list and checks if it is able to treat the content of the action list</li> <li>5. The AXMEDIS Client Application runs all actions that it can execute directly</li> <li>6. The AXMEDIS Client Application forwards to other applications (explicitly indicated with the action to execute) all actions that it cannot execute directly</li> <li>7. The AXMEDIS Object reaches its final destination</li> </ol>
<b>Post-conditions</b>	Typical actions are copy, move, play, apply the AXMEDIS Object.
<b>Variations</b>	Actions on the AXMEDIS Object could be applied before the end of transmission.
<b>Asynchronous actions</b>	The loss of satellite signal in particular weather conditions
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

## 12.6 Content Protection for Satellite distribution

<b>UCId</b>	UC11.6
<b>Use case</b>	Content Protection for Satellite distribution
<b>Description</b>	During the Satellite Data Broadcast the AXMEDIS Object is further protected at transport level
<b>Actors</b>	The AXMEDIS B2B Distributor.
<b>Assumptions</b>	Transport uses TCP protocol encapsulated in the DVB-MPE standard
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Client Application identifies an incoming AXMEDIS Object like encrypted content</li> <li>2. The AXMEDIS Client Application launches the Application to decrypt the incoming packets, using the Conditional Access System (CAS) developed internally by EUTELSAT.</li> <li>3. Encrypted packets of AXMEDIS Object are sent to the 'Decrypting Box' for decrypting packets</li> <li>4. Decrypted packets of AXMEDIS Object are assembled by the AXMEDIS Client Application in order to re-build the original Object</li> </ol>
<b>Post-conditions</b>	The AXEMDIS Object should respect the DRM rules, even when the AXMEDIS Object has been rebuilt.
<b>Variations</b>	The 'Decrypting Box' is represented either by an internal software component or by an external component (e.g., smart card reader)
<b>Asynchronous actions</b>	The 'Decrypting Box' could have some problems and stop receiving encrypted packets
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

## 13 AXMEDIS Protection Tools

### 13.1 Super AXCS

#### 13.1.1 AXMEDIS Registration of AXCSs

<b>UCId</b>	UC13.1.1
<b>Use case</b>	AXMEDIS Registration of AXCSs
<b>Description</b>	An actor wants to register an AXCS in the AXMEDIS system
<b>Actors</b>	AXCS manager (a B2B user managing AXCS)
<b>Assumptions</b>	The AXCS to be registered is already installed on a machine
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXCS Client Registration Application (related to the AXCS to be registered) is started by the Actor</li> <li>2. The AXCS contacts the AXMEDIS Registration of AXCSs Web Service providing all the data required for the registration in the system.</li> <li>3. The AXMEDIS Registration of AXCSs Web Service verifies the received data and answer to the requesting AXCS providing a new ID and other needful data. It verifies also the integrity of the AXCS.</li> <li>4. the AXCS stores the received data</li> </ol>
<b>Post-conditions</b>	The requesting AXCS is registered in the system
<b>Variations</b>	If the data received by AXMEDIS Registration of AXCSs Web Service is rejected the requesting AXCS is not registered in the system and a communication is sent to the requesting Actor
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.1.2 Tool/device off-line registration

<b>UCId</b>	UC13.1.2
<b>Use case</b>	Tool/device off-line registration
<b>Description</b>	An Actor wants to register a new kind of tool in the AXMEDIS network
<b>Actors</b>	AXMEDIS tool producer (i.e. a software house producing a specified tool to use it in the AXMEDIS system )
<b>Assumptions</b>	The tool is not already registered in the system
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Reception of the tool that wants to be registered in the AXMEDIS system</li> <li>2 Off-line checking and test that tool accomplishes AXMEDIS guidelines</li> <li>3 If the tool accomplishes AXMEDIS guidelines <ol style="list-style-type: none"> <li>3.1 The tool is registered and certified, i.e. tool fingerprint is estimated and other major parameters are extracted for verification at each transactions.</li> <li>3.2 The tool is registered in the AXCS Registration and Certification Database with all the information collected at the step 3.1 using the AXMEDIS SW Tools off-line Registration</li> </ol> </li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The tool is registered in the AXCS Registration and Certification Database</li> <li>• A new tool type id is generated and bounded to the tool type</li> <li>• The requester receives notification about the registration</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.1.3 AXMEDIS Object ID Generation

**13.1.3.1 Generation of unique Object ID**

<b>UCId</b>	UC13.1.3.1
<b>Use case</b>	Generation of unique object ID
<b>Description</b>	An actor wants to associate an AXMEDIS Object ID to the newly created object.
<b>Actors</b>	Integrator, Designer
<b>Assumptions</b>	AXMEDIS Editor is opened (or tool using AXOM)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an object creator requests a “New object ID” in the user interface</li> <li>2 AXOM sends request to PMS Client (DRM support component) to get authorisation for the operation</li> <li>3 DRM Support component of PMS sends operation request to AXMEDIS Registrator Web Service together with information on the object which the identifier is requested for</li> <li>4 AXMEDIS Registrator Web Service generates the OID with respect to the information received.</li> <li>5 The OID is returned to the AXOM by DRM Support</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	To have OID does not mean that the objects has been protected. In fact the object does not have the final shape and thus its final fingerprint. When the object is protected the final fingerprints have to be sent to the AXCS for storing them into the database associated to the AXOID, etc. The fingerprint can be at level of single resource and thus the Resource ID internal to the object is needed. In addition a global fingerprint for the whole object could be estimated.

**13.1.3.2 Registration of metadata about a new object**

<b>UCId</b>	UC13.1.3.2
<b>Use case</b>	Registration of metadata about a new object
<b>Description</b>	An actor wants to register metadata about a new AXMEDIS Object.
<b>Actors</b>	Integrator, Designer
<b>Assumptions</b>	The actor has already requested and obtained a new unique object ID (see use case 13.1.3.1); AXMEDIS Editor is opened (or tool using AXOM)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an object creator requests for a new object metadata registration in the user interface</li> <li>2 AXOM sends request to PMS Client (DRM support component) to get authorisation for the operation</li> <li>3 DRM Support component of PMS sends operation request to AXMEDIS Registrator Web Service together with information on the object which the identifier is requested for</li> <li>4 AXMEDIS Registrator Web Service registers information received.</li> <li>5 The OID is returned to the AXOM by DRM Support</li> </ol>
<b>Post-conditions</b>	Object metadata are stored in the AXCS Objects ID database
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	To register object metadata does not mean that the object has been stored in the AXMEDIS Database (AXDB). In fact, the object itself is not involved in the registration process, it neither gets transferred to the AXCS; the registration process described involves only object metadata. The real object storing involves directly the AXDB, not the AXCS.

### 13.1.4 Global Object List WEB Service

#### 13.1.4.1 Search of AXMEDIS Objects

<b>UCId</b>	UC13.1.4.1
<b>Use case</b>	Search of AXMEDIS Objects
<b>Description</b>	An Actor wants to perform a search in the AXCS database to retrieve a set of AXMEDIS Objects satisfying several conditions
<b>Actors</b>	Any user
<b>Assumptions</b>	The search can be performed “by hand” using the web interface provided by the service or using an AXMEDIS Tool. In the first case the query is composed using the web interface, in the latter case the query is composed inside the tool.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Actor contacts the Global Object List WEB Service using an AXMEDIS Tool or a Web browser. In the following statements the program used to interact with the Global Object List WEB Service is referred as “Client”</li> <li>2. The Actor compose the query using the client</li> <li>3. The query is submitted to the Global Object List WEB Service</li> <li>4. The Global Object List WEB Service contacts the AXCS Database Interface to submit the received query</li> <li>5. The AXCS Database Interface perform the query over the database and sends the retrieved data to the Global Object List WEB Service</li> <li>6. The Global Object List WEB Service sends the received data to the requesting client</li> </ol>
<b>Post-conditions</b>	The Client receives a list of AXMEDIS Objects (according with the sent query)
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Web service
<b>Issues</b>	It has to be remarked AXCS does not contain Objects themselves, but only metadata about objects. Real AXMEDIS Objects are retained by Distributors, and in general by AXMEDIS farms

### 13.1.5 Super AXCS Collector

#### 13.1.5.1 On-line transfer among AXCSs and SuperAXCSs

<b>UCId</b>	UC13.1.5.1
<b>Use case</b>	On-line transfer among AXCSs and Super AXCSs
<b>Description</b>	Some information managed by AXCS during an AXMEDIS Object usage has to be transferred among AXCSs and SuperAXCSs. This transfer involves AXCS Synchronizer and AXCS Collector
<b>Actors</b>	End user
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 End user uses an AXMEDIS tool to operate on AXMEDIS Protected Objects that are on different distribution channels</li> <li>2 Protection Manager Support allows only authorized operations on the objects</li> <li>3 Objects are accessed on different channels and each AXCS stores its Action-Logs</li> <li>4 Via AXCS synchronizer general information on Objects, Users and Reporting is transferred among the AXCS and SuperAXCS network using AXCS Collector</li> <li>5 AXCSs and SuperAXCSs collect and store the received information</li> </ol>
<b>Post-conditions</b>	Object usage information are transferred among AXCSs and SuperAXCSs network
<b>Variations</b>	4a. If some node is not reachable, AXCS synchronizer stores the information to be transferred in a queue called AXCS Synchronizer Queue. Information stored in that queue is transferred to AXCS Collector when the connection returns active.

<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

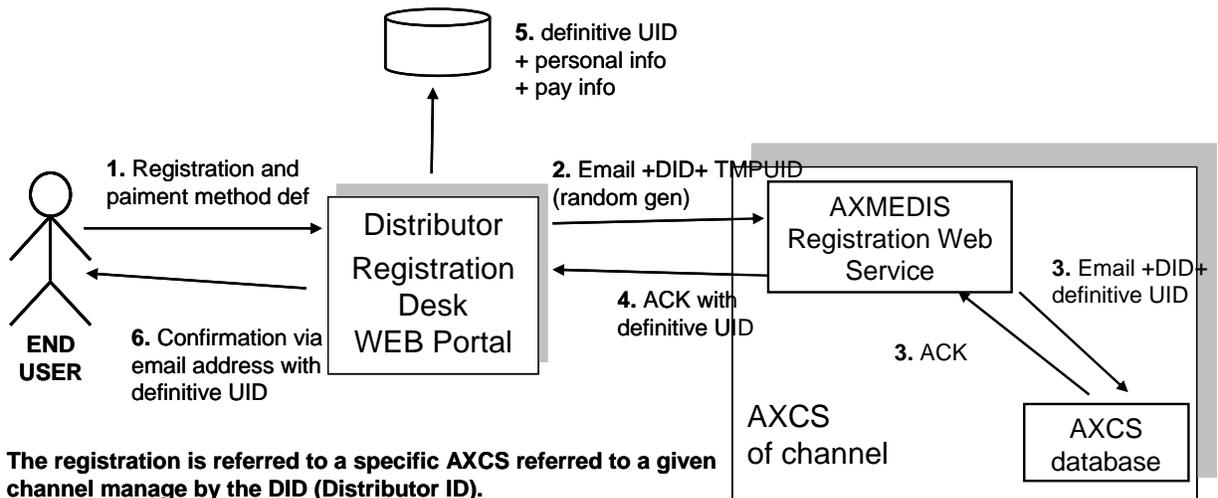
**13.1.5.2 Off-line synchronization among AXCSs and SuperAXCSs**

<b>UCId</b>	UC13.1.5.2
<b>Use case</b>	Off-line synchronization among AXCSs and Super AXCSs
<b>Description</b>	Some information collected by AXCS during an AXMEDIS Object usage has to be transferred among AXCSs and SuperAXCSs even if the connection among AXCSs and SuperAXCSs is interrupted. In this case the transfer doesn't occur on-line during the Object usage, but off-line in a second time. This use case describes the line-up among the AXCS databases and the SuperAXCS databases.
<b>Actors</b>	Super AXCS Collector
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Super AXCS Collector retrieves the list of all the AXCS registered in the system (performing a query to Active AXCS List Database)</li> <li>2. Super AXCS Collector slides every entry in that list and, for each AXCS, sends a Queue Pull Request, i.e. a request for data present in the AXCS Synchronizer Queue.</li> <li>3. The contacted AXCS Synchronizer respond providing the requested data (if present) and empty the AXCS Synchronizer Queue.</li> </ol>
<b>Post-conditions</b>	Object usage information is transferred among the AXCS and SuperAXCS network
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.2 AXMEDIS Certifier and Supervisor**

**13.2.1 AXMEDIS Registration Service**

**13.2.1.1 End User registration in a distribution channel**



Instead of a definitive UID we can use a "Certificate" or what we can call the AXMEDIS Personal Identity Card (AXPIC). It can be a certificate that one can exhibit to authenticate himself/herself in the AXMEDIS circuit, a check is typically done with that ID and the email ,etc...

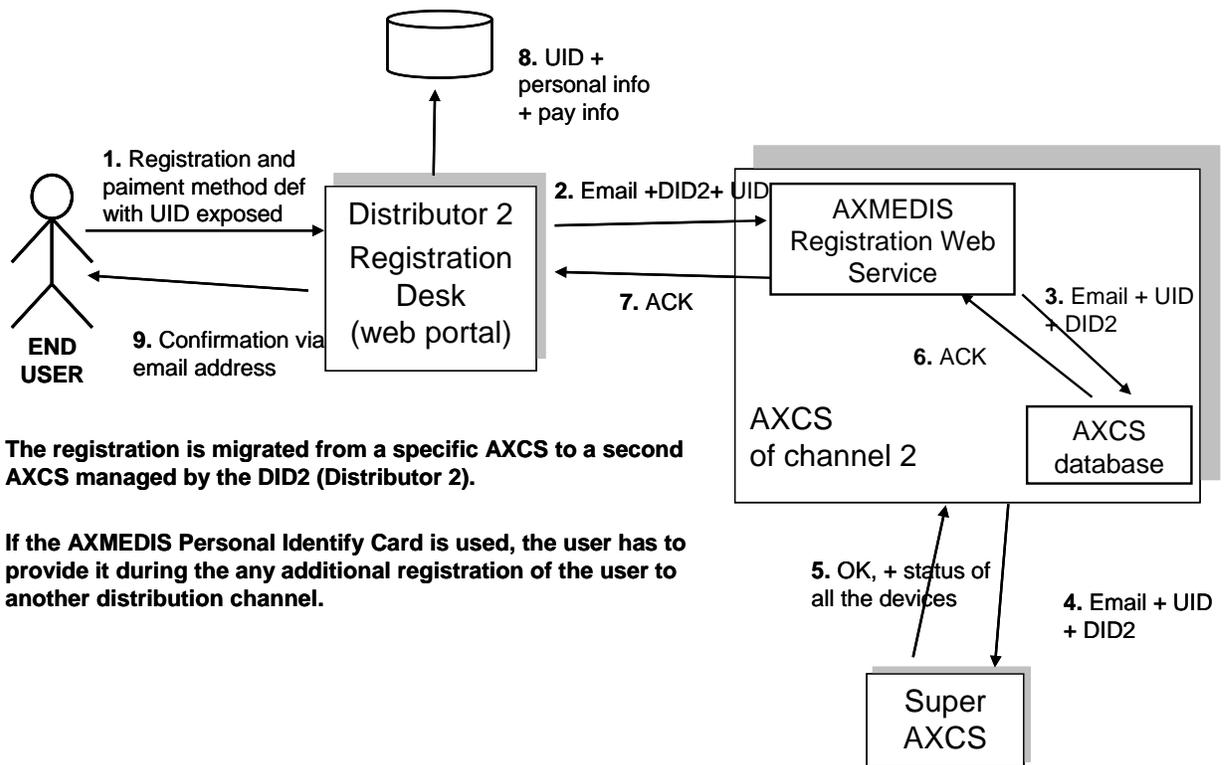
<b>UCId</b>	UC13.2.1.1
<b>Use case</b>	User registration in a distribution channel
<b>Description</b>	An actor wants to register in a channel
<b>Actors</b>	All AXMEDIS Users (we can include: End User, Distributor, Content Provider, Collecting Society and so on)
<b>Assumptions</b>	Distributor supports directly new AXMEDIS User registration and stores directly user personal data
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor registers his/her data together with the payment method</li> <li>2 Some non personal Actor's data (such as email, nick name, password, etc.) and DID (Distributor ID) are transferred to the AXMEDIS registration service</li> <li>3 The AXMEDIS User ID (AXUID) along with an appropriate certificate are generated and acknowledged by AXCS</li> <li>4 An acknowledge with the certificate (containing the definitive ID) is sent back to the distributor</li> <li>5 The associations between Actor's data and the definitive ID are stored into distributor user database</li> <li>6 A confirmation email is sent back to the Actor</li> </ol>
<b>Post-conditions</b>	The user is registered in the AXCS Registration and Certification Database
<b>Variations</b>	5a – In case the distributor doesn't want to stores associations between user personal data and AXUID, step 5 doesn't take place. In successive transactions, AXUID is communicated to the distributor by the tool, which automatically establish an SSL connection to the distributor site based on the user certificate.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### **13.2.1.2 End User registration in AXMEDIS system without a referencing distributor**

<b>UCId</b>	UC13.2.1.2
<b>Use case</b>	End User registration in AXMEDIS system without a referencing distributor
<b>Description</b>	An actor wants to register in a channel
<b>Actors</b>	All AXMEDIS Users (we can include: End User, Distributor, Content Provider, Collecting Society and so on)
<b>Assumptions</b>	The actor does not want to choose a referencing distributor or no distributor directly supports new user registration. The user has an AXMEDIS compliant tool (called "tool"). The registration process is carried out through the tool.

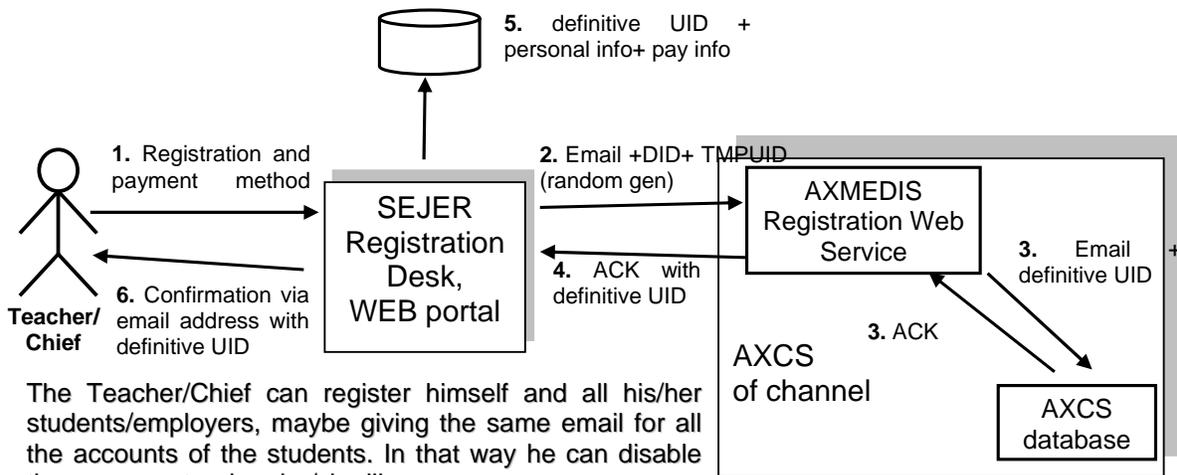
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor registers his/her data together with the payment method using the tool which establishes a connection to the AXMEDIS User Registration Portal</li> <li>2 The portal temporarily stores user data and sends back a confirmation email</li> <li>3 The user click on a link contained in the confirmation email and confirms his registration in the appropriate web page</li> <li>4 Some non personal Actor's data (such as email, nick name, password, etc.) and DID (Distributor ID) are transferred to the AXMEDIS registration service</li> <li>5 The AXMEDIS User ID (AXUID) along with an appropriate certificate are generated and acknowledged by AXCS</li> <li>6 An acknowledge with the certificate (containing the definitive ID) is sent back to the portal</li> <li>7 The portal consolidates user personal data</li> <li>8 The associations between Actor's data and the definitive ID are stored into portal user database</li> <li>9 The certificate is sent back to the Actor's tool which automatically stores it</li> </ol>
<b>Post-conditions</b>	The user is registered in AXMEDIS, called an AXMEDIS User
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	In this use case the user has no referencing distributor, so distributors do not know the associations between user personal data and AXUIDs. In successive transactions, AXUID is communicated to the distributor by the tool, which automatically establish an SSL connection to the distributor site based on the user certificate.

**13.2.1.3 End User registration in a different distribution channel**



<b>UCId</b>	UC13.2.1.3
<b>Use case</b>	User registration in a second channel
<b>Description</b>	An actor wants to register in a second channel
<b>Actors</b>	End User
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor registers his/her data together with the payment method. UID is sent to the system</li> <li>2 The Actor's email, DID and UID are transferred to the AXMEDIS registration service</li> <li>3 The AXMEDIS User ID in conjunction with new DID and email is sent to AXCS</li> <li>4 The AXMEDIS User ID in conjunction with new DID and email is sent also to the Super AXCS</li> <li>5 Super AXCS acknowledge</li> <li>6 AXCS acknowledge</li> <li>7 AXMEDIS registration service acknowledges the request</li> <li>8 User ID is store with other user data in the Distributor user database</li> <li>9 The Actor receives confirmation of the registration to the distribution channel</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.2.1.4 Registration of a new structured group of people**



The Teacher/Chief can register himself and all his/her students/employers, maybe giving the same email for all the accounts of the students. In that way he can disable those accounts when he/she likes.

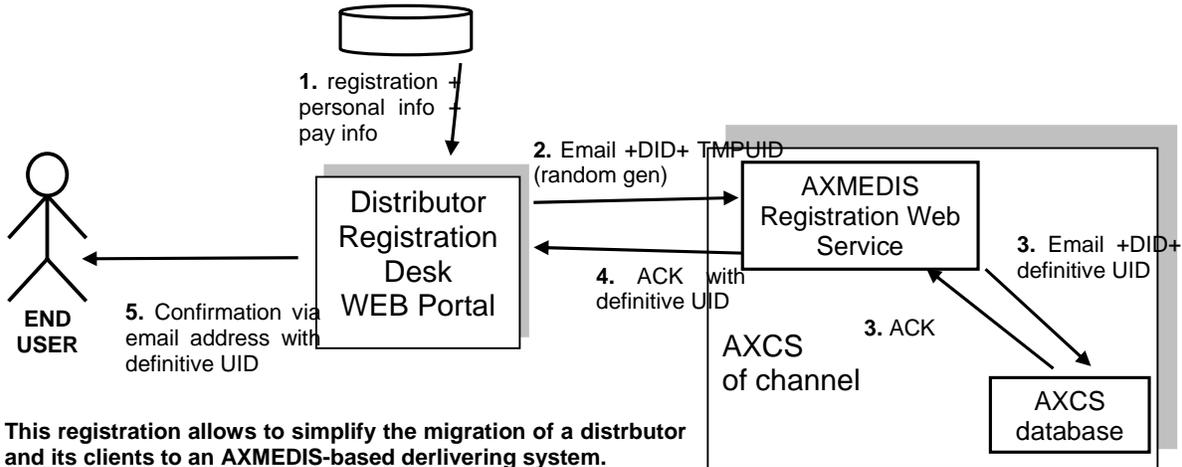
- Chief: <UID354135>
- Person sdhfg: <UID134514>
- Person afsdhKLFH: <UID675737>
- Person dgag: <UID437673>
- Person rtywuyert: <UID36773673>

Different UID (or AXMEDIS Personal Identify Cards) will be received by the Teacher, that has to save all the info (emails)

UCId	UC13.2.1.4
------	------------

<b>Use case</b>	Registration of a new structured group of people
<b>Description</b>	An Actor wants to register a structured group of people in the AXMEDIS network
<b>Actors</b>	Chief of a structured group composed by some kind of people (such as Teacher/Students, Chief/Employers and so on), AXCS, Distributor or AXMEDIS Registration Portal
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The Actor registers his/her data together with the payment method to the Distributor Portal or to the AXMEDIS Registration Portal</li> <li>2. Some non personal Actor's data (such as email, nick name, password, etc.) and DID (Distributor ID) are transferred to the AXMEDIS registration service</li> <li>3. The AXMEDIS User ID (AXUID) along with an appropriate certificate are generated and acknowledged by AXCS</li> <li>4. An acknowledge with the certificate (containing the definitive ID) is sent back to the distributor/AXMEDIS Registration Portal</li> <li>5. The associations between Actor's data and the definitive ID are stored into distributor/AXMEDIS Registration Portal user database</li> <li>6. A confirmation email is sent back to the Actor</li> <li>7. Steps 1 through 6 are performed by the Actor for each user of the structured group</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• All the people belonging to the structured group are registered in the AXCS Registration and Certification Database</li> <li>• All the people belonging to the structured group receive notification about the registration</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.2.1.5 Registration of an old User of the Channel on AXMEDIS**



This registration allows to simplify the migration of a distributor and its clients to an AXMEDIS-based delivering system.

Instead of a definitive UID we can use a "Certificate" or what we can call the AXMEDIS Personal Identity Card (AXPIC). It can be a certificate that one can exhibit to authenticate himself/herself in the AXMEDIS circuit, a check is typically done with that ID

<b>UCId</b>	UC13.2.1.5
<b>Use case</b>	Registration of an old User of the Channel on AXMEDIS

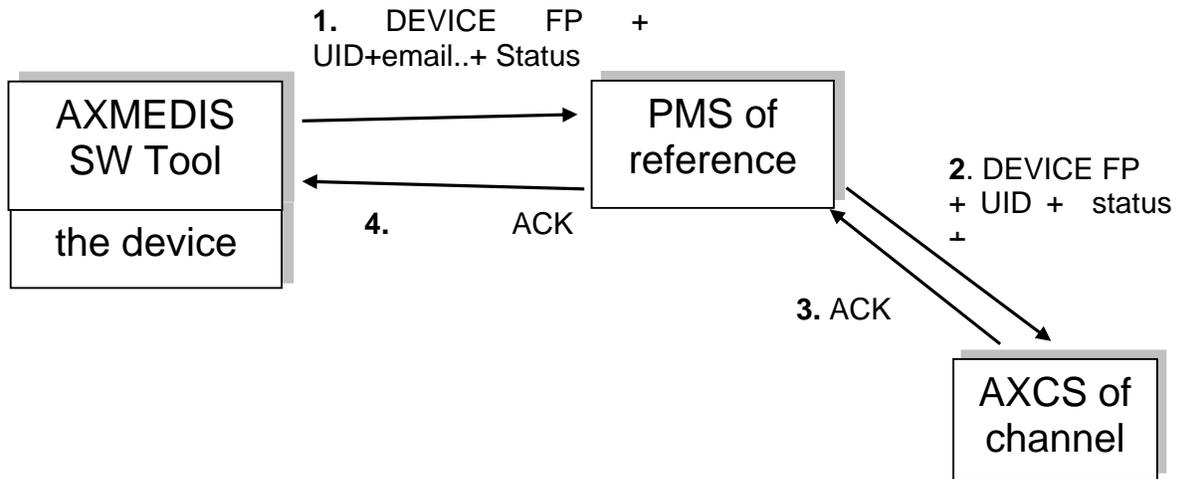
<b>Description</b>	An Actor wants to register an End User in the AXMEDIS network. This use case is indented to facilitate the migration of Distributors of Channel to the AXMEDIS system
<b>Actors</b>	Distributor
<b>Assumptions</b>	Distributor supports directly AXMEDIS User registration and stores directly user personal data
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Distributor retrieves the User's information from its database</li> <li>2 The Distributor sends a registration request to the service filled with non personal data about the user to be registered in the system (see mentioned data in use case "End User registration in a distribution channel")</li> <li>3 The service checks and validates the data received. Then the service generate a user id, a login, a password and a certificate for the new user</li> <li>4 The service sends to the requester the generated certificate</li> <li>5 The User receives confirmation of the registration from the Distributor along with the certificate</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The user is registered in the AXCS Registration and Certification Database</li> <li>• The User receives notification about the registration and user id, login, password and certificate</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.1.6 User password modification

<b>UCId</b>	UC13.2.1.6
<b>Use case</b>	User password modification
<b>Description</b>	An Actor wants to change a user password
<b>Actors</b>	Content Provider, Distributor
<b>Assumptions</b>	The user is already registered in the system
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor sends a password modification request to the service filled with the user id, old password, new password</li> <li>2 The service checks and validates the data received</li> <li>3 The service updates information related to the user in the database and change the user password as specified</li> <li>4 The service sends to the requester the confirmation of the password modification</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The user password is stored in the AXCS Registration and Certification Database</li> <li>• The requester receives notification about the password modification</li> </ul>
<b>Variations</b>	Distributor portal can be substituted with AXMEDIS Registration Portal
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

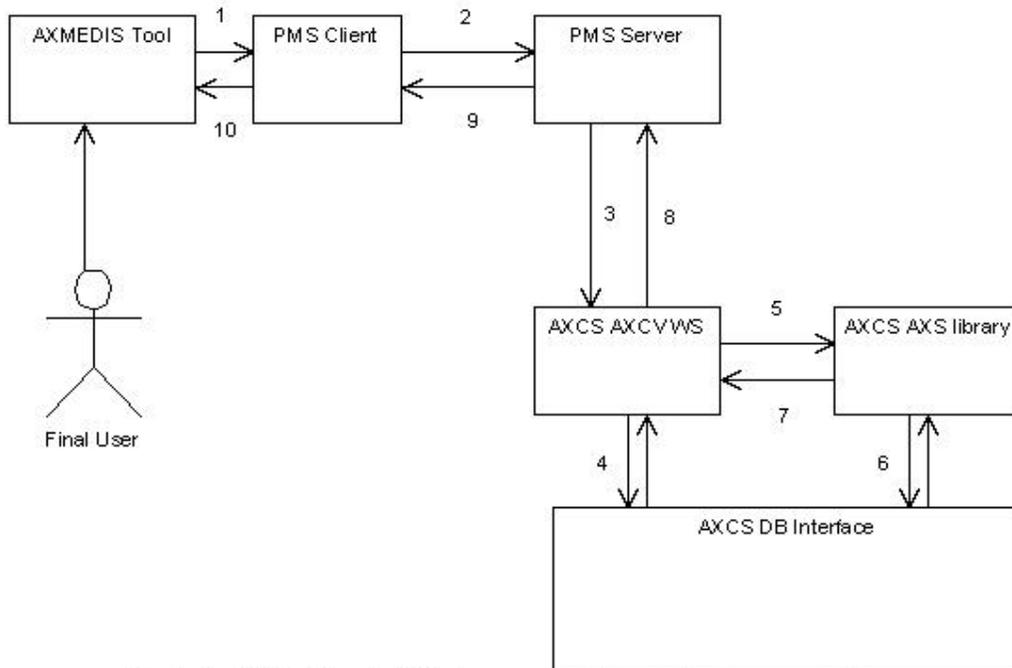
### 13.2.2 AXMEDIS Certification and Verification

#### 13.2.2.1 Authentication of a Tool and a Device



<b>UCId</b>	UC13.2.2.1
<b>Use case</b>	Authentication of a Tool and a Device
<b>Description</b>	The Device/Tool needs to be authenticated while it communicates with PMS. The authentication can be performed every gap of time, requested by PMS, AXCS, AXOM (or others subjects) or in any other way it is convenient. This use case show how the authentication occurs, not when nor why.
<b>Actors</b>	AXMEDIS tool running on a device
<b>Assumptions</b>	The specified tool has been already certified
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Tool starts connection with the PMS of reference sending him some information such as DEVICE FP, UID, email, Status and so on.</li> <li>2. The PMS sends the received information to the AXCS of channel (eventually adding other)</li> <li>3. The AXCS (the AXMEDIS Certification and Verification) verifies the received information (in particular the status is important) and sends the response to the PMS</li> <li>4. The mentioned PMS sends the response to the AXMEDIS Tool: in this way the chain is closed</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	If the AXMEDIS Certification and Verification does not authenticate the tool and device, it will block it in the system and return an error code that must be interpreted by AXOM to deactivate immediately the tool in the client side.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.2.2.2 Certification of AXMEDIS Tool by a User on a Device



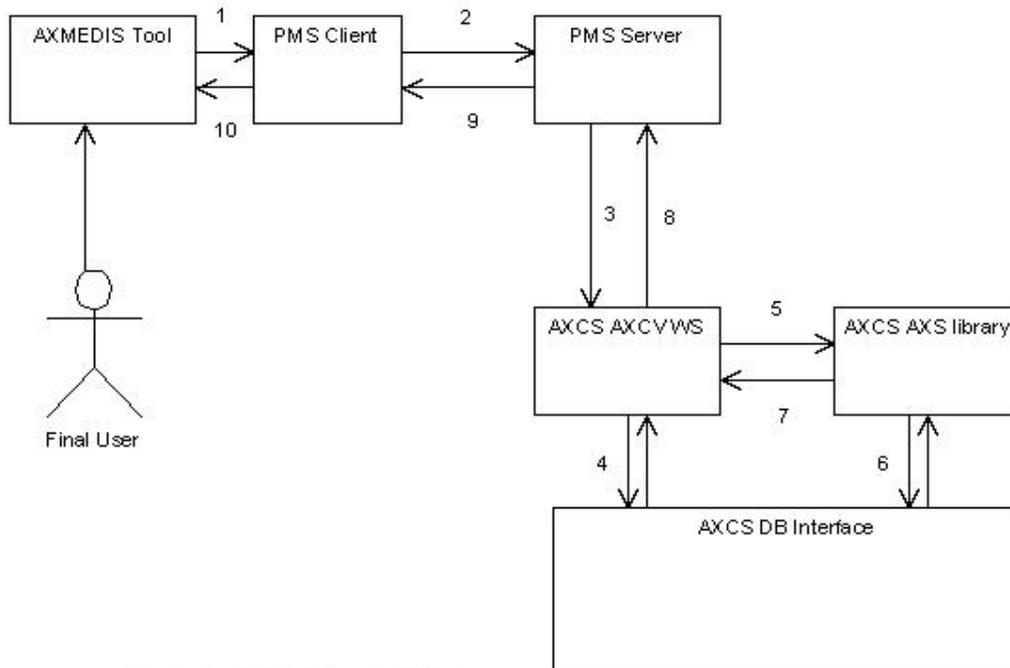
A Final User wants to use a tool for the first time

1. The tool uses PMS Client to send the PMS Server the following information: AXUID, AXRTID, tool Fingerprint, etc. in order to certify the tool.
2. PMS Client retrieves and adds domain information and contacts PMS Server
3. PMS Server contacts AXCS-AXCV
4. AXCS-AXCV checks the user status and tool integrity and determines that the tool was not already certified. AXC V generates AXTID, tool certificate and enabling code and creates a new entry in AXCS RegCert database.
5. AXC V sends AXS a SupervisorInputData (SID) which indicates that the user has certified a tool on a device.
6. AXS stores SID in the AXCS database
7. AXS confirms the storage
8. AXC V returns the result of the operation, and if the certification is successful it also returns AXTID, tool certificate and enabling code
9. PMS Server passes the received information to PMS Client
10. PMS Client gives the information to AXMEDIS Tool

<b>UCId</b>	UC13.2.2.2
<b>Use case</b>	Certification of AXMEDIS tool by a user on a Device
<b>Description</b>	An Actor wants to certify a specific tool installed on a terminal (i.e. a PC, a Palmtop, a Phone, a Kiosk and so on) that is used for the first time
<b>Actors</b>	AXMEDIS User (can be end user, distributor or whoever uses a tool), AXMEDIS Tool, AXCS Database Interface, PMS
<b>Assumptions</b>	A User wants to use an AXMEDIS tool The specified tool is not already certified: it's the first time it is used

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the tool for its certification</li> <li>2 AXOM (as a part of the tool) calculates fingerprint and extracts other features to identify the specified tool, the user and the terminal it is installed on</li> <li>3 AXOM (as a part of the tool) contacts the pertinent PMS sending all the needful information for the certification: AXUID, AXRTID, tool fingerprint, registration deadline.</li> <li>4 The mentioned PMS contacts the pertinent AXMEDIS Certification and Verification sending him all the received information plus AXDOM</li> <li>5 AXCXV checks that the tool was not previously certified.</li> <li>6 AXCXV checks if the user and tool are registered (check that the user data and status are correct, that the received Tool FP matches the original one and that the registration deadline does not exceed the maximum tool deadline). AXMEDIS Certification and Verification generates a TID (tool ID), an enabling code and a tool private key and x509 certificate and inserts it together with all the received information in the AXCS Database (except the private key), using the proper interface. At the end, the tool has been certified in AXMEDIS by the corresponding user.</li> <li>7 AXMEDIS Certification and Verification sends to PMS the generated TID, enabling code and a PKCS12 structure that includes the tool private key and certificate.</li> <li>8 The PMS sends to AXOM (as a part of the tool) a confirmation message, including previous information.</li> <li>9 AXOM stores the received TID, imports the certificate and private key and enables the tool.</li> <li>1 If the user or tool are not registered, the PMS and AXOM are sent a message notifying the unsuccessful certification</li> <li>10 If some information about the tool fingerprint or deadline is not correct, the user and registered tool are blocked and a SupervisorInputData is stored in AXCS accounting database to explain the reason why.</li> </ol>
<b>Post-conditions</b>	<p>If the user and tool are registered, and the received information is correctly verified</p> <ul style="list-style-type: none"> <li>• The tool is certified in the AXMEDIS system</li> <li>• A new tool ID, enabling code and certificate are generated and bounded to the tool</li> <li>• The requester receives notification about the certification</li> </ul>
<b>Variations</b>	If the tool is registered but the user is not, the user might be asked to register
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.2.3 Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption



A Final User wants to consume an AXMEDIS object on a tool that has been already certified in AXMEDIS:

1. The tool uses PMS Client to send the PMS Server the following information: AXUID, AXTID, tool Fingerprint Digest (includes software and device hardware and installation information), list of performed actions to be resynchronized, etc. in order to verify user and tool status and integrity
2. PMS Client retrieves and adds domain information and contacts PMS Server
3. PMS Server contacts AXCS-AXCV
4. AXCS-AXCV checks the user status, tool certification in AXMEDIS and tool integrity
5. AXCS-AXCV sends AXCS-AXS the list of performed actions
6. AXCS-AXS checks that the list of actions is consistent in terms of fingerprint history
7. AXCS-AXS notifies AXCS-AXCV the successful storage
8. AXCV notifies PMS Server the successful verification
9. PMS Server passes the received information to PMS Client
10. PMS Client gives the information to AXMEDIS Tool

See as a reference the previous figure.

<b>UCId</b>	<b>UC13.2.2.3</b>
<b>Use case</b>	<b>Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption</b>
<b>Description</b>	Verify the user data, tool data and tool operation history consistency before a user wants to use an AXMEDIS tool not for the first time and/or every time the Tool is connected
<b>Actors</b>	AXMEDIS Tool, PMS, AXCS, AXCS Database Interface
<b>Assumptions</b>	The AXMEDIS Tool (AXOM) verifies the user and tool data before requesting authorisation to perform an action.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Tool (AXOM) verifies the user and tool data before requesting authorisation to perform an action.</li> <li>2. AXMEDIS Tool (AXOM) sends some needful information to PMS, such as: AXUID, AXTID and the tool fingerprint digest (hardware and software).</li> <li>3. PMS of reference contacts the Domain Manager to retrieve the domain the user is working in (AXDOM).</li> <li>4. PMS of reference contacts AXCS-AXCV sending the received information plus the AXDOM (domain) and the list of operations performed during offline operation (list of ActionLogs)</li> <li>5. AXCV verifies the received user and tool information.</li> <li>6. If AXCS-AXCV doesn't verify correctly the tool fingerprint digest, it returns an error code an AXOM must then try to reverify the tool sending he whole tool fingerprint instead of the digest. See "Reverification of AXMEDIS users using AXMEDIS tools" use case.</li> <li>7. Certification and verification (AXCV, inside AXCS) checks that the tool is certified. Otherwise it returns a notification message. See "Certification of AXMEDIS User and Tool" use case</li> <li>8. AXCV contacts AXMEDIS Supervisor (AXS) to verify and store the list of offline operations. AXS recalculates and verifies for each received Action Log the tool history fingerprint againsta the one that is stored in the AXCS database (using AXCS Database interface).</li> <li>9. If all the calculated fingerprints match the received fingerprint present in the the corresponding Action Log, the PMS is notified that the user data, tool data and operation history have been verified.</li> </ol>
<b>Post-conditions</b>	<p>If AXCS-AXCV doesn't verify the tool operations history hash, the certified tool is blocked and it should be immediately deactivated by AXOM when it receives the "unsuccessful reverification" notification message.</p> <p>If the verification is performed during online consumption and the tool operation history hash is not correctly verified, the user is blocked.</p> <p>When a tool or user are blocked, a SupervisorInputData is stored in AXCS accounting database to explain the reason why.</p>
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.2.4 Reverification of AXMEDIS users using AXMEDIS tools on a Device during content consumption inside a domain

See as a reference the same figure of the previous scenarios.

<b>UCId</b>	<b>UC13.2.2.4</b>
<b>Use case</b>	<b>Reverification of AXMEDIS users using AXMEDIS tools</b>
<b>Description</b>	Reverify the user data, tool data and tool operation history consistency before a user wants to use an AXMEDIS tool not for the first time and/or every time the Tool is connected, when the verification has failed because of tool fingerprint digest mismatch
<b>Actors</b>	AXMEDIS Tool, PMS, AXCS, AXCS Database Interface
<b>Assumptions</b>	The AXMEDIS Tool (AXOM) has previously tried to perform the "Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption inside a domain" scenario, and it has failed due to a tool fingerprint digest mismatch

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The AXMEDIS Tool (AXOM) verifies the user and tool data before requesting authorisation to perform an action.</li> <li>2. AXMEDIS Tool (AXOM) sends some needful information to PMS, such as: AXUID, AXTID and whole tool fingerprint (hardware and software).</li> <li>3. PMS of reference contacts the AXCS sending him the received information plus the list of operations performed during offline operation (list of ActionLogs)</li> <li>4. AXCv verifies the received user and tool information.</li> <li>5. Certification and verification (AXCV, inside AXCS) checks that the tool is certified. Otherwise it returns a notification message. See “Certification of AXMEDIS User and Tool” use case</li> <li>6. AXCv contacts AXMEDIS Supervisor (AXS) to verify and store the list of offline operations. AXS recalculates and verifies for each received Action Log the tool history fingerprint againsta the one that is stored in the AXCS database (using AXCS Database interface).</li> <li>7. If all the calculated fingerprints match the received fingerprint present in the corresponding Action Log, the PMS is notified that the user data, tool data and operation history have been verified.</li> </ol>
<b>Post-conditions</b>	<p>If AXCS-AXCV doesn’t reverify correctly the tool fingerprint or the tool operations history hash, the certified tool is blocked and it should be immediately deactivated by AXOM when it receives the “unsuccessful reverification” notification message.</p> <p>If the reverification is performed during online consumption and the tool operation history hash is not correctly verified, the user is blocked.</p> <p>When a tool or user are blocked, a SupervisorInputData is stored in AXCS accounting database to explain the reason why.</p>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.3 Manual Blocking

#### 13.2.3.1 Manual User Blocking/Unblocking

<b>UCId</b>	UC13.2.3.1
<b>Use case</b>	User blocking/unblocking
<b>Description</b>	An Actor wishes to block/unblock a user in the AXMEDIS network
<b>Actors</b>	Any B2B user interested in blocking/unblocking (Distributors, Content Providers, Collecting Societies, etc), AXCS Management Console Administrator
<b>Assumptions</b>	The user to be (un)blocked is registered in the system. The (un)blocking request has been sent by an entitled authority
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The block/unblock requester sends a block/unblock request to an authority entitled to decide for (un)blocking</li> <li>2 The authority checks and validates the received request</li> <li>3 If the check is passed, the authority sends the request to the proper AXCS administrator</li> <li>4 The AXCS administrator blocks/unblocks the user using the AXCS Management Console</li> <li>5 The AXCS administrator sends to the requester the confirmation of the (un)blocking of the user</li> </ol>

<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>The user is marked as (un)blocked in the AXCS Registration and Certification Database (its own status results changed)</li> <li>The requester receives notification about the (un)blocking</li> </ul>
<b>Variations</b>	If the check performed at step 3 is not passed, the requester receives a bad request notification
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.3.2 Certified Tool blocking/unblocking

The “registration” term refers to Tool Off-line Registration scenario. A registered tool is a software product. An instance of a Registered Tool running on a terminal becomes a Certified Tool.

Blocking a tool can have different “rules”:

- Blocking a specific certified tool belonging to one user (e.g. due to manipulations).
- Blocking a specific version of the tool (named registered tool) in a mandatory manner (e.g. that specific version has a security hole and an exploit has been released over the internet). It is a way to force downloading a new version to preserve system integrity.

<b>UCId</b>	UC13.2.3.2
<b>Use case</b>	Certified tool blocking/unblocking
<b>Description</b>	An Actor wishes to block/unblock a certified tool in the AXMEDIS network
<b>Actors</b>	Any B2B user interested in blocking/unblocking (Distributors, Content Providers, Collecting Societies, etc), AXCS Management Console Administrator
<b>Assumptions</b>	The tool to be (un)blocked is certified in the system. The (un)blocking request has been sent by an entitled authority
<b>Steps</b>	<ol style="list-style-type: none"> <li>The block/unblock requester sends a block/unblock request to an authority entitled to decide for (un)blocking</li> <li>The authority checks and validates the received request</li> <li>If the check is passed, the authority sends the request to the proper AXCS administrator</li> <li>The AXCS administrator blocks/unblocks the certified tool using the AXCS Management Console</li> <li>The AXCS administrator sends to the requester the confirmation of the (un)blocking of the certified tool</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>The certified tool is marked as (un)blocked in the AXCS Registration and Certification Database (its own status results changed)</li> <li>The specific instance of a tool (named certified tool) results (un)blocked for all users. Other tool instances belonging to the same tool family (named registered tool) are not affected by the status change</li> <li>The requester receives notification about the (un)blocking</li> </ul>
<b>Variations</b>	If the check performed at step 3 is not passed, the requester receives a bad request notification
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.3.3 Registered Tool blocking/unblocking

<b>UCId</b>	UC13.2.3.3
-------------	------------

<b>Use case</b>	Registered tool blocking/unblocking
<b>Description</b>	An Actor wishes to block/unblock a registered tool in the AXMEDIS network
<b>Actors</b>	Any B2B user interested in blocking/unblocking (Distributors, Content Providers, Collecting Societies, etc), AXCS Management Console Administrator
<b>Assumptions</b>	The tool to be (un)blocked is registered in the system. The (un)blocking request has been sent by an entitled authority
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The block/unblock requester sends a block/unblock request to an authority entitled to decide for (un)blocking</li> <li>2 The authority checks and validates the received request</li> <li>3 If the check is passed, the authority sends the request to the proper AXCS administrator</li> <li>4 The AXCS administrator blocks/unblocks the registered tool using the AXCS Management Console</li> <li>5 The AXCS administrator sends to the requester the confirmation of the (un)blocking of the registered tool</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The registered tool is marked as (un)blocked in the AXCS Registration and Certification Database</li> <li>• All the certified tool instances belonging to that tool family (named registered tool) result (un)blocked for all users</li> <li>• The requester receives notification about the (un)blocking</li> </ul>
<b>Variations</b>	If the check performed at step 3 is not passed, the requester receives a bad request notification
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.4 AXMEDIS Supervisor

#### 13.2.4.1 AXMEDIS Protection Information delivery

<b>UCId</b>	UC13.2.4.1
<b>Use case</b>	AXMEDIS Protection Information delivery
<b>Description</b>	AXMEDIS protection Information delivery for supporting consumption of AXMEDIS objects
<b>Actors</b>	PMS Server, AXS
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 PMS Server requests AXCS-AXS the protection information associated to an AXMEDIS object</li> <li>2 AXMEDIS Supervisor verifies the request validity</li> <li>3 AXS retrieves the information through AXCS database Interface and, if present, returns it to PMS.</li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• PMS Server will perform the license-based authorisation and store the corresponding ActionLog or SupervisorInputData before it returns the protection information to PMS Client</li> <li>•</li> </ul>
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Please note that into the protection information is contained. Key, algorithms/rules, etc., for descrambling, or other information, for the protection processor, etc.

**13.2.4.2 Storage/update of protection information of an AXMEDIS Object to the AXCS**

<b>UCId</b>	UC13.2.4.2
<b>Use case</b>	Storage/update of protection information of an AXMEDIS Object to the AXCS
<b>Description</b>	Protection information are stored and associated to a given AXMEDIS Object
<b>Actors</b>	PMS Server, AXS
<b>Assumptions</b>	An AXMEDIS User has protected an AXMEDIS object and generated the related protection information
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 AXMEDIS Supervisor receives a store/update request containing the AXOID of the protected object, the version, the protection timestamp and the related protection information</li> <li>2 AXMEDIS Supervisor verifies the request validity</li> <li>3 The protection information is stored in AXCS database and linked to the given AXOID.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.2.4.3 Storage of Action Logs in AXCS Accounting database**

<b>UCId</b>	UC13.2.4.3
<b>Use case</b>	Storage of Action Logs in AXCS Accounting database
<b>Description</b>	Storage of reports of performed actions (Action Logs) through AXMEDIS Supervisor
<b>Actors</b>	PMS, AXCS-AXCV, AXCS-AXS
<b>Assumptions</b>	
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. PMS Server receives a list of Action Logs or generates an Action Log after authorising</li> <li>2. PMS Server sends AXCS-AXS the list of one or more Action Logs to be stored</li> <li>3. AXCS-AXS calculates and verifies the history consistence for each received Action Log, it sets the HistVerSuccess field and the registration timestamp to the appropriate value and stores them in AXCS accounting database.</li> <li>4. If the history is not consistent, AXCS-AXS blocks the tool and stores an SupervisorInputData in AXCS Accounting database that explains the reasons.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.2.4.4 Storage of SupervisorInputData in AXCS Accounting database**

<b>UCId</b>	UC13.4.4
<b>Use case</b>	Storage of SupervisorInputData in AXCS Accounting database
<b>Description</b>	Storage of reports about some specific events in AXMEDIS system
<b>Actors</b>	PMS, AXCS-AXCV, AXCS-AXS

<b>Assumptions</b>	PMS Server generates a SupervisorInputData when someone requests a license and when the result of an authorisation is negative. AXCV and AXS generate a SupervisorInputData when a tool is certified in the system and when user or a tool is blocked in the system.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. AXCS-AXS receives a SupervisorInputData coming from PMS Server, AXCS-AXCV or AXCS-AXS</li> <li>2. AXCS-AXS verifies it has the necessary information stores the SupervisorInputData in AXCS Accounting database.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 13.2.5 AXMEDIS Reporting and Statistics Web Service

### 13.2.5.1 Object usage reporting for accounting purposes

<b>UCId</b>	UC13.2.5.1
<b>Use case</b>	Object usage reporting for accounting purposes
<b>Description</b>	An Actor wants data about object usage for accounting purposes
<b>Actors</b>	Business users interested in accounting activities (e.g. Collecting Societies, Distributors, etc.), CAMART, AXCS
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A business users interested in accounting activities sends a report request to the CAMART providing its own id, a timestamp and some other criteria</li> <li>2 The service checks and validates the received data</li> <li>3 The service collects information related to the requestor pertinent object usage from AXCS using the Reporting Web Service and sends it back to the requester</li> </ol>
<b>Post-conditions</b>	The Actor has the report
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	CAMART is the only tool dealing with Reporting Web Service

### 13.2.5.2 Object usage reporting for statistics purposes

<b>UCId</b>	UC13.2.5.2
<b>Use case</b>	Object usage reporting for statistics purposes
<b>Description</b>	An Actor wants data about object usage for statistics purposes
<b>Actors</b>	Business users interested in statistic activities (e.g. Collecting Societies, etc.), CAMART, AXCS
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A business users interested in statistic activities sends a report request to the CAMART providing its own id, a timestamp and some other criteria</li> <li>2 The service checks and validates the received data</li> <li>3 The service collects anonymous information about object usage from AXCS using the Statistics Web Service and sends it back to the requester</li> </ol>
<b>Post-conditions</b>	The Actor has the report
<b>Variations</b>	None

<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	CAMART is the only tool dealing with Statistics Web Service

### 13.2.6 Accounting Manager and Reporting Tool

#### 13.2.6.1 List of all operations performed on an object

<b>UCId</b>	UC13.2.5.1
<b>Use case</b>	List of all operations performed on an object
<b>Description</b>	An Actor wants to know all operations performed on an object
<b>Actors</b>	Content provider, distributor, collecting society
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 An Actor sends the request for having the list of all operation of an object: the request contain the Object ID and the code of the operation to be performed, and time window related to the period</li> <li>2 The service checks and validates the data received</li> <li>3 The service collects and sends back information related to the object in the database</li> </ol>
<b>Post-conditions</b>	The Actor has the list
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.2.6.2 List of all operations performed by a user

<b>UCId</b>	UC13.2.5.2
<b>Use case</b>	List of all operations performed by a user
<b>Description</b>	An Actor wants to know all operations performed by a user
<b>Actors</b>	Distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 An Actor sends the request for having the list of all operation performed by an user: the request contain the User ID, the Actor ID and the code of the operation to be performed, and time window related to the period</li> <li>2 The service checks and validates the data received</li> <li>3 The service collects and sends back Action-Logs related to the user</li> </ol>
<b>Post-conditions</b>	The Distributor has the list
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.2.6.3 Usage report about an object

<b>UCId</b>	UC13.2.5.3
<b>Use case</b>	Usage report about an object
<b>Description</b>	An Actor wants to know usage statistic about an object
<b>Actors</b>	Distributor, Collecting society, content owner, content provider
<b>Assumptions</b>	None

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 An Actor sends the request for having the statistic about an object usage: the request contain the Object ID, the Actor ID and the code of the operation to be performed, and time window related to the period</li> <li>2 The service checks and validates the data received</li> <li>3 The service collects and sends back information related to the object usage</li> </ol>
<b>Post-conditions</b>	The Actor obtains the statistic
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### **13.2.6.4 Usage report about a distributor**

<b>UCId</b>	UC13.2.5.4
<b>Use case</b>	Usage report about a distributor
<b>Description</b>	An Actor wants to know usage statistic about a distributor
<b>Actors</b>	Distributor, Collecting society, content owner, content provider
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor sends a request to the service: the request contains the Distributor ID, the Actor ID and the operation to be performed, and time window related to the period</li> <li>2 The service checks and validates the data received</li> <li>3 The service collects and sends back information related to the distributor</li> </ol>
<b>Post-conditions</b>	The Actor obtains the statistic
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### **13.2.6.5 Usage report about a provider**

<b>UCId</b>	UC13.2.5.5
<b>Use case</b>	Usage report about a provider
<b>Description</b>	An Actor wants to know usage statistic about a provider
<b>Actors</b>	Distributor, Collecting society, content owner, content provider
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor sends a request to the service: the request contains the Provider ID, the Actor ID and the operation to be performed, and time window related to the period</li> <li>2 The service checks and validates the data received</li> <li>3 The service collects and sends back information related to the provider</li> </ol>
<b>Post-conditions</b>	The Actor obtains the statistic
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### **13.2.6.6 List objects for which an administrative account can be requested**

<b>UCId</b>	UC13.2.5.6
<b>Use case</b>	List objects which an administrative account can be requested

<b>Description</b>	An Actor wants to obtain and consult the list of objects for which an administrative account can be requested, e.g. all the objects for which the Actor has the eligibility to obtain an administrative report)
<b>Actors</b>	Content creators, distributors, collecting society, content providers
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 An Actor requests the list of objects for which an administrative account can be requested</li> <li>2 The reporting tool searches in the corresponding database all the AXMEDIS objects for which the actor is eligible to ask a report.</li> <li>3 The reporting tool lists all the results.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.6.7 Listing AXMEDIS clients of a distributor/channel

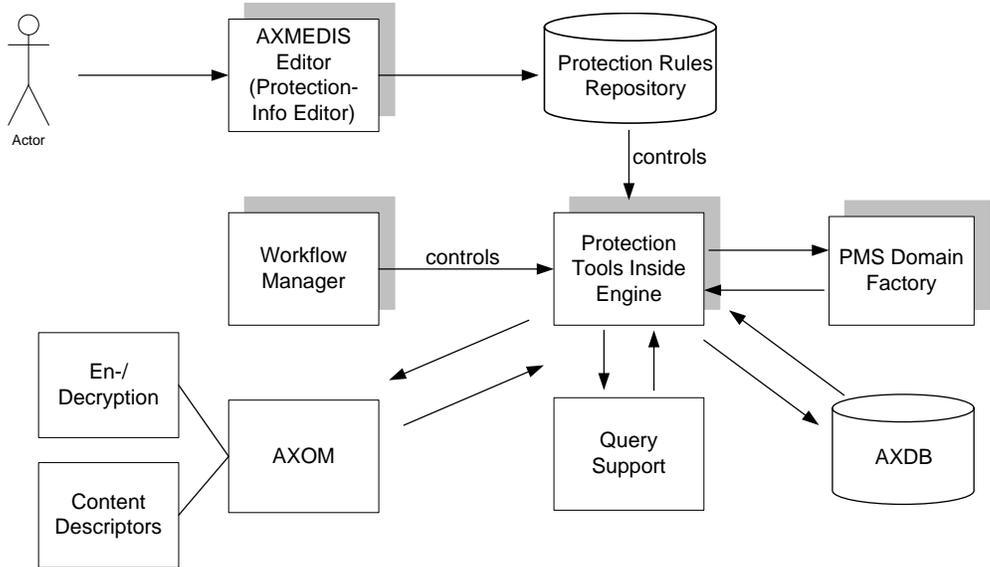
<b>UCId</b>	UC13.2.5.7
<b>Use case</b>	Listing all AXMEDIS clients
<b>Description</b>	An Actor wants to consult the AXMEDIS clients list that has been connected to the distributor or on a channel.
<b>Actors</b>	Distributors.
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor asks for the list of AXMEDIS clients that have been connected to a distributor of a channel</li> <li>2 The reporting tool searches in the corresponding database all the AXMEDIS clients satisfying point 1.</li> <li>3 The reporting tool lists all the results.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.2.6.8 Listing distributors

<b>UCId</b>	UC13.2.5.8
<b>Use case</b>	Listing all distributors of AXMEDIS objects, those that have redistributed its objects.
<b>Description</b>	An Actor wants to consult the distributors of AXMEDIS objects.
<b>Actors</b>	Content creators, Distributors, End users, Content Providers.
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 An actor asks for a list of distributors of AXMEDIS objects</li> <li>2 The reporting tool searches in the corresponding database all the distributors of AXMEDIS objects</li> <li>3 The reporting tool lists all the results.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3 Protection Tool Engine

Protection tool engine use cases regarding DRM support are defined in section 13.5 Protection Manager Support / Server.



#### 13.3.1 Content protection

<b>UCId</b>	UC13.3.1
<b>Use case</b>	Content protection
<b>Description</b>	An AXMEDIS object is protected. The Protection tools allow the protection of a large set of objects when they are reused into the AXCP. Here, protection includes adaptation of PAR and in a second phase also the creation of Licenses. Licenses are created according to the given rules. The right to create a specific license is evaluated by considering the PARs. Also, objects are connected with Licenses resulting in a Governed AXMEDIS Object.
<b>Actors</b>	Content creator and content distributor.
<b>Assumptions</b>	AXMEDIS object exists and is uniquely identified.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Protection Tool Engine is initiated to protect an object. This can be done via the AXCP or the Protection Information Editor (with the Protection Rules Repository).</li> <li>2 Parameters are verified and rule is loaded</li> <li>3 Content is selected via the Query Support.</li> <li>4 Content is accessed via the AXOM.</li> <li>5 The PAR are estimated on the basis of the licenses of the included resources and then the PAR is included in the AXInfo contained into the protected and non protected parts of the object.</li> <li>6 PMS Domain factory creates/adaptes the license from the rules or the user input.</li> <li>7 Verification of PAR or License against given rights</li> <li>8 PMS Domain factory creates required keys (e.g. for encryption or hash functions).</li> <li>9 Creation of the protection information</li> <li>10 Protection the object (resulting in a new object or a new version of the object). Encryption support (see use case Encryption) is used via the AXOM.</li> <li>11 If the protection is successful and the protection information has been generated to protect this object, the protection information has to be stored (see use case Storage of security information)</li> <li>12 Sending the license and the Protection information to the PMS. The PMS forwards the Protection Information to the AXCS.</li> </ol>
<b>Post-conditions</b>	AXMEDIS object(s) is (are) encrypted and stored in the AXMEDIS database together with relevant information
<b>Variations</b>	<p>The request can originate from different sources, e.g.</p> <ul style="list-style-type: none"> <li>• Protection Info Editor (AXMEDIS Editor)</li> <li>• via the AXOM, e.g. from the AXEPTool</li> <li>• via the WF-Manager: This allows the automatic protection of content after its production by the Formatting Tools Engine.</li> </ul> <p>Different parameters can be given:</p> <ul style="list-style-type: none"> <li>• Selection of AXMEDIS object(s) and all protection parameters explicitly</li> <li>• Selection of AXMEDIS object(s) and rules</li> </ul> <p>This is the general case including:</p> <ul style="list-style-type: none"> <li>• Rule based protection of single and multiple objects</li> <li>• Automatic protection during publication via AXEPTool or other tools (e.g. editor)</li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.2 Create a new protection rule

<b>UCId</b>	UC13.3.2
<b>Use case</b>	Create a new protection rule
<b>Description</b>	An Actor wants to create a new protection rule, that include a Protection Information model
<b>Actors</b>	Content creator and content distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Actor creates a Selection of digital resources by making queries to the AXMEDIS Database</li> <li>2 The Actor rules how these resources have to be protected</li> <li>3 The Actor stores the created rule into Protection rules Database[FHGIGD: Where are the protection rules stored?]</li> </ol>

<b>Post-conditions</b>	None
<b>Variations</b>	1) The Actor defines a Selection by writing in the rule the scripting code (Protection rule Language) for queries to be executed when the rule will be run 2) The Actor can define a rule or writing it as scripting code (Protection rule Language) or in a Visual way.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.3 Search and Select a protection rule

<b>UCId</b>	UC13.3.3
<b>Use case</b>	Search and Select a protection rule
<b>Description</b>	An Actor wants to Select a specific protection rule he should be enabled to make some search or browsing.
<b>Actors</b>	Content creator and content distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>5 The Actor searches in the Protection rules database for a specific protection rule</li> <li>6 If the Actor finds the rule he can : <ol style="list-style-type: none"> <li>6.1 Use it to create a protected AXMEDIS object</li> <li>6.2 Modify it <ol style="list-style-type: none"> <li>6.2.1 Then the Actor store the new rule into the Repository by Protection tool user interface and rule editor</li> <li>6.2.2 Use the new rule to create a protected AXMEDIS object</li> </ol> </li> </ol> </li> <li>7 If the Actor does not find the rule, he can create a new one</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.4 Activating a protection rule

<b>UCId</b>	UC13.3.4
<b>Use case</b>	Activating a protection rule
<b>Description</b>	An Actor wants to activate a protection rule
<b>Actors</b>	Content creator and content distributor
<b>Assumptions</b>	The Protection tool user interface and rule editor can access the Protection Rules Database
<b>Steps</b>	<ol style="list-style-type: none"> <li>5 The Actor searches in the Protection rules database for a specific protection rule</li> <li>6 If the Actor does not find the rule or a new model for Protection Information <ol style="list-style-type: none"> <li>6.1 The Actor can create a new one</li> </ol> </li> <li>7 The Actor selects “Activate Rule” function</li> <li>4 The rule is put as active rule into the Protection Rules Database</li> </ol>
<b>Post-conditions</b>	The rule is added to the set of Active Rule
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.3.5 Removing a protection rule**

<b>UCId</b>	UC13.3.5
<b>Use case</b>	Removing a protection rule
<b>Description</b>	An Actor wants to remove a protection rule
<b>Actors</b>	Content creator and content distributor
<b>Assumptions</b>	The Protection tool user interface and rule editor can access the Protection Rules Database
<b>Steps</b>	10 The Actor requests the list of Active Rules in the Protection Rules Database 11 The Actor selects the active rule to be disabled 12 The Actor selects “Remove Rule” function 13 The rule is Removed
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.3.6 Debugging a protection rule**

<b>UCId</b>	UC13.3.6
<b>Use case</b>	Debugging/Simulation a protection rule
<b>Description</b>	An Actor wants to debug a protection rule
<b>Actors</b>	Content creator and content distributor
<b>Assumptions</b>	A protection rule is available.
<b>Steps</b>	4 The Use Case starts when the Actor wants to debug a rule 5 The Rule Editor enters in the Debugging/Simulation Mode 6 During the debugging mode the Actor can: 6.1 Check the statements of rule step by step 6.2 Control the values of current variables 6.3 Exit from the debugging mode
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.3.7 Editing protection rules**

<b>UCId</b>	UC13.3.2
<b>Use case</b>	Editing protection rules
<b>Description</b>	A user creates or edits a protection rule.
<b>Actors</b>	Content creator and content distributor..
<b>Assumptions</b>	AXMEDIS object exists and is uniquely identified.
<b>Steps</b>	1 User opens a new or an existing protection rule. 2 User adds, edits, or removes new rules. 3 Protection rule is stored.
<b>Post-conditions</b>	New or modified rules are stored
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Rules are created with a Protection Tool Editor and executed by the Protection Tool Engine. Rules are stored in a separate database or file system and have to be ready to be activated and executed and searched by the Protection Tool Editor.

<b>Issues</b>	None
---------------	------

### 13.3.8 Printing protection rules

<b>UCId</b>	UC13.3.8
<b>Use case</b>	Content protection
<b>Description</b>	A user prints the protection rule and related protection information
<b>Actors</b>	Content creator and content distributors
<b>Assumptions</b>	eules exists.
<b>Steps</b>	1 The user requests to print the protection rules. 2 Protection rules are printed.
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

The following aspects and use cases have not been produced in time and will be produced for the next version.

From the WF is possible to control both the Editor and the Engine. This allows passing from Formatting to Protection according to some defined flow.

The Protection Tool has to:

- protect AXMEDIS objects in large set, adapting the DRM, PAR, Licenses, etc.
- merge a license and an object to create a Governed AXMEDIS Objects
- generate licenses in automatic according to some rules

The Protection Tool Engine is an instance of the AXCP Engine with specific support to work with protection functionalities

The Protection Tool Editor is an instance of the AXCP Engine with specific support to work with protection functionalities

### 13.3.9 Automated creation and association of licences

<b>UCId</b>	UC13.3.9
<b>Use case</b>	Automated creation and association of licences
<b>Description</b>	A user wants to created licences automatically for <ul style="list-style-type: none"> <li>• Different AXMEDIS objects</li> <li>• Different customers</li> </ul> The licences are automatically associated with the corresponding AXMEDIS objects.
<b>Actors</b>	AXMEDIS Object creator, AXCP Engine
<b>Assumptions</b>	The user is logged in and his identity has been validated.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user creates the relevant data to generate the licence (e.g. principal, resource, granted rights and conditions (limitations). The parameters are not fixed to a single value (to allow the creation of multiple licenses/for multiple objects).</li> <li>2. The Licence Generator creates the licence.</li> <li>3. The Licence Verifier validates the licence. <ol style="list-style-type: none"> <li>a. If validation is successful: <ul style="list-style-type: none"> <li>• licence is stored in the database (DRM licence)</li> <li>• the licence ID is returned</li> <li>• the licence ID is associated with the new AXMEDIS object</li> </ul> </li> <li>b. If validation failed: <ul style="list-style-type: none"> <li>• feedback to the user is given (depending on the user)</li> </ul> </li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<ul style="list-style-type: none"> <li>• Licence related data is derived from the licence model (based on the licence model ID)</li> <li>• Licence related data is derived from the PAR</li> <li>• The AXMEDIS content (or parts thereof) is encrypted before generating the corresponding license, related functionalities: <ol style="list-style-type: none"> <li>i. encryption (see UC Encryption)</li> <li>ii. storage of protection information (see UC Storage of security information)</li> </ol> </li> </ul>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.10 Automated verification of licences or PARs

<b>UCId</b>	UC13.3.10
<b>Use case</b>	Automated verification of licences or PARs
<b>Description</b>	The user can verify if certain rights and conditions <ol style="list-style-type: none"> <li>1. of an existing object are given</li> <li>2. of a non existing object can be granted</li> </ol>
<b>Actors</b>	AXMEDIS Object creator, AXCP Engine
<b>Assumptions</b>	The user is logged in and his identity has been validated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user creates the relevant data that has to be verified (e.g. principal, resource, granted rights and conditions (limitations).</li> <li>2. The user creates the data for the selection of the object</li> <li>3. The rights are validated against the licence</li> <li>4. Feedback to the user is given (depending on the user)</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Instead of testing possible rights against the licence the user can test the right against the PAR (e.g. as a test before creating the final (end) user license.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.11 Automated editing of PARs

<b>UCId</b>	UC13.3.11
<b>Use case</b>	Automated editing of PARs
<b>Description</b>	For multiple objects the PARs can be modified by the AXCP according to the given information

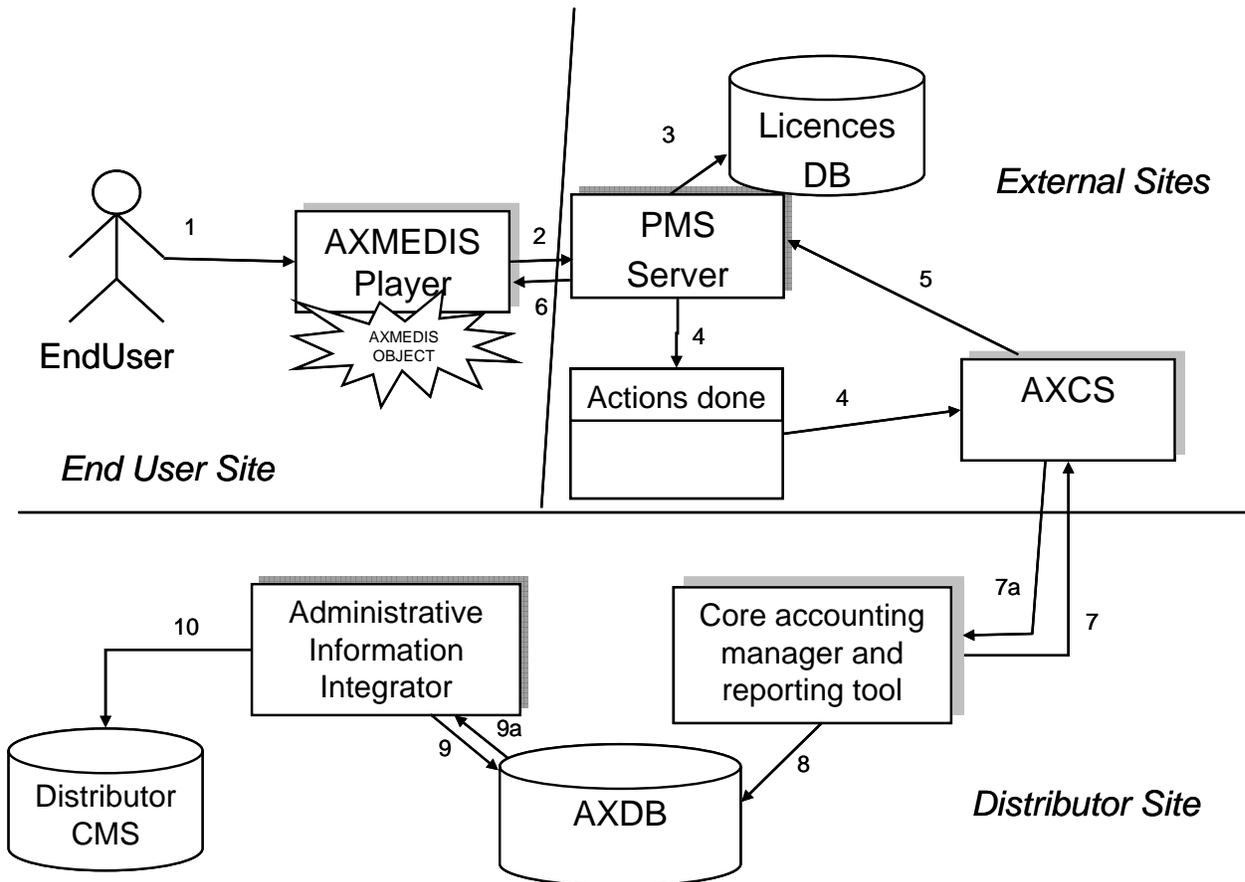
<b>Actors</b>	AXMEDIS Object creator, AXCP Engine
<b>Assumptions</b>	The user is logged in and his identity has been validated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user creates the relevant data that has to be verified (e.g. principal, resource, granted rights and conditions (limitations)).</li> <li>2. The user creates the data for the selection of the object where the PAR should be modified</li> <li>3. The PARs are modified (for each object)</li> <li>4. Feedback to the user is given (depending on the user)</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.3.12 Automated editing of licences

<b>UCId</b>	UC13.3.12
<b>Use case</b>	Automated editing of licences
<b>Description</b>	For multiple objects the licences can be modified by the AXCP according to the given information
<b>Actors</b>	AXMEDIS Object creator, AXCP Engine
<b>Assumptions</b>	The user is logged in and his identity has been validated.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user creates the relevant data that has to be modified (e.g. principal, resource, granted rights and conditions (limitations)).</li> <li>2. The user creates the data for the selection of the object</li> <li>3. The licences are modified</li> <li>4. Feedback to the user is given (depending on the user)</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.4 Administrative Information Integrator

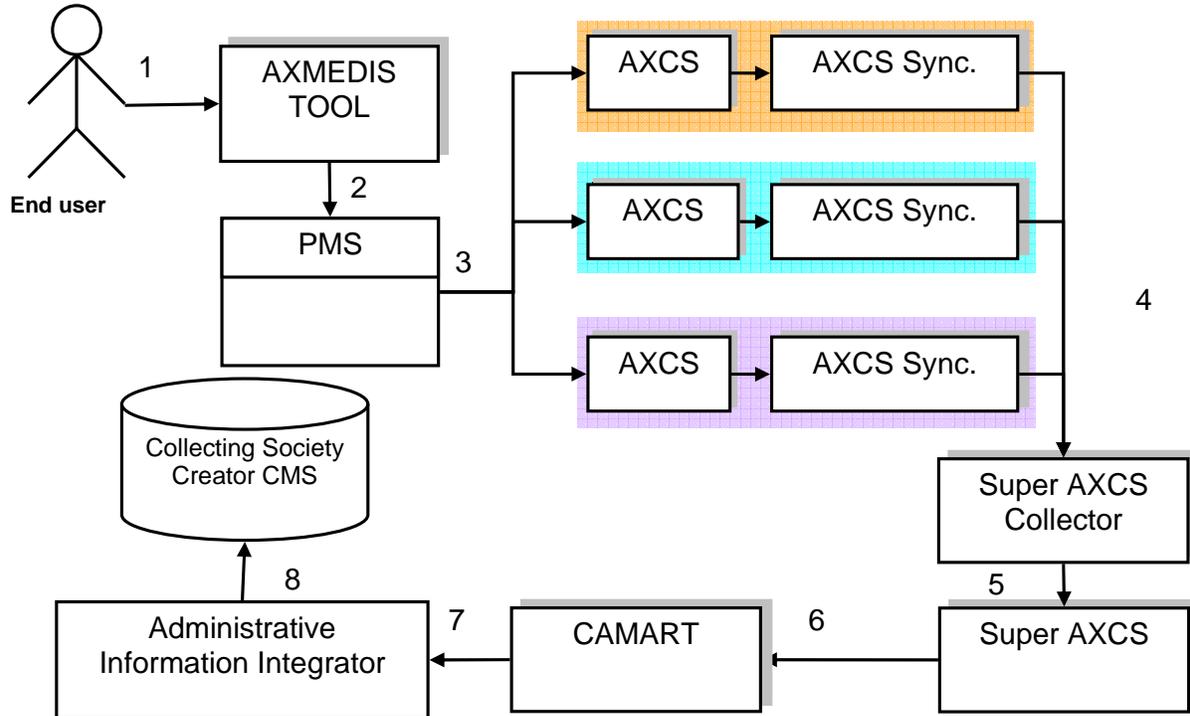
#### 13.4.1 Integrating Distributor administrative information of the basis of End User actions



<b>UCId</b>	UC13.4.1
<b>Use case</b>	Integrating Distributor administrative information of the basis of End User actions
<b>Description</b>	In response to an user action, the administrative information are transferred to the CMS of distributor
<b>Actors</b>	End user
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. End User requests to perform an action on an AXMEDIS Protected Object</li> <li>2. AXMEDIS Player asks PMS to perform an Action (assuming client has been already certified)</li> <li>3. PMS checks in the LicenceDB if the Action is allowed (assuming OK)</li> <li>4. PMS sends AXCS the action performed</li> <li>5. AXCS gives back the key to access the content (if necessary)</li> <li>6. PMS gives the grant to access the content and possibly the key to the AXMEDIS Player</li> <li>7. Accounting &amp; Rep. Tool retrieves from AXCS the actions performed by all the End Users on objects distributed by the distributor</li> <li>8. A&amp;R Tool stores the transactions in the AXDB</li> <li>9. Adm. Integrator gets transactions performed from the DB</li> <li>10. Administrative information are mapped into the Distributor CMS</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None

<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

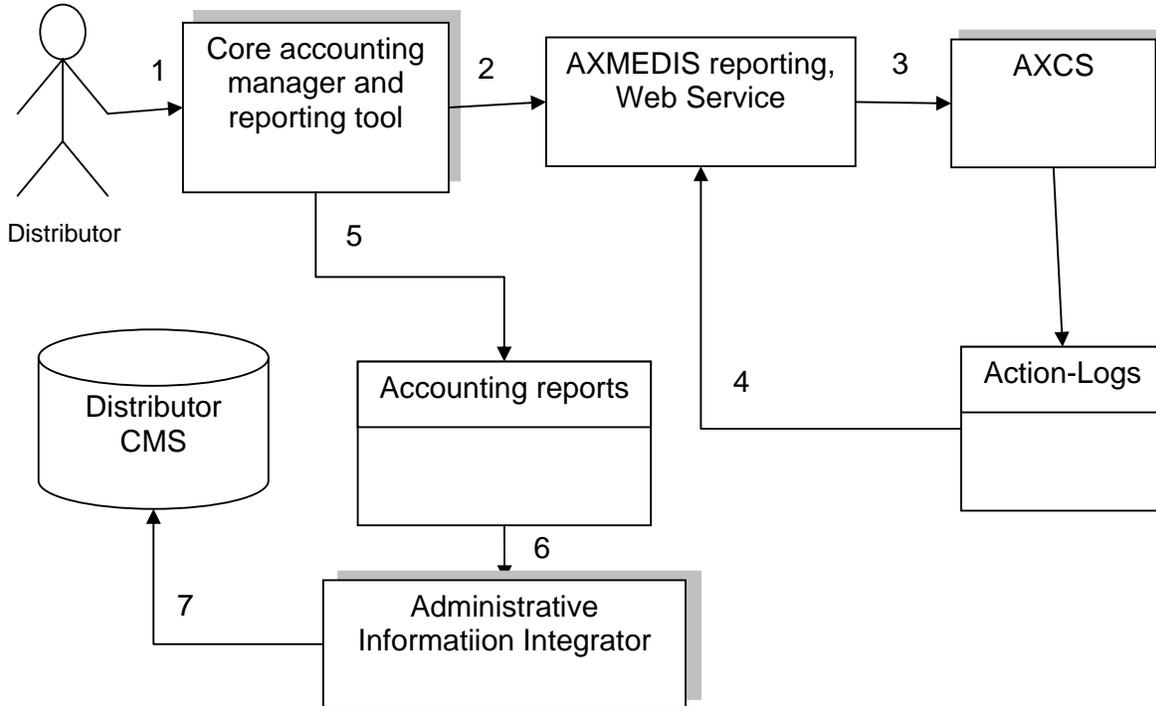
**13.4.2 Integrating Collecting Society administrative information of the basis of End User actions**



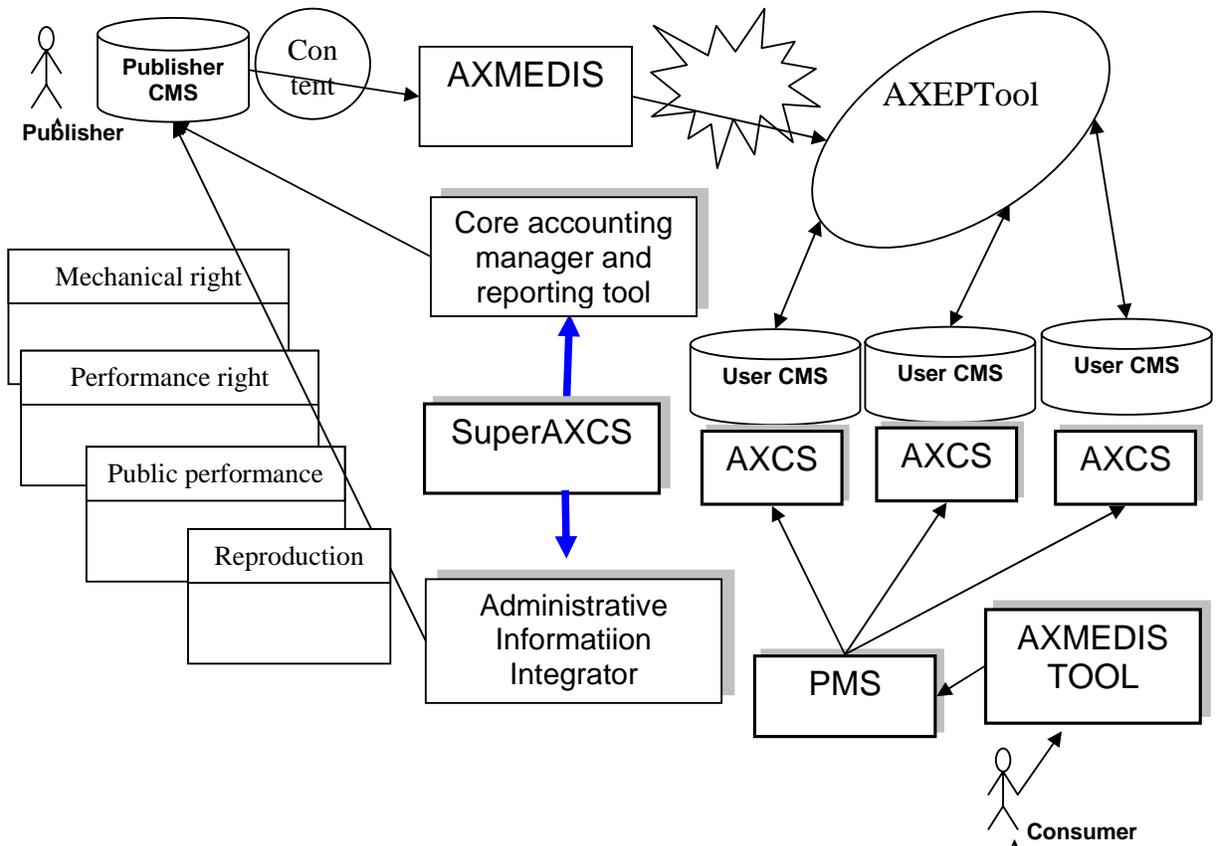
<b>UCId</b>	UC13.4.2
<b>Use case</b>	Integrating Collecting Society administrative information of the basis of End User actions
<b>Description</b>	In response to an user action, the administrative information are transferred to the CMS of collecting society
<b>Actors</b>	End user
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. End user uses an AXMEDIS tool to operate on an AXMEDIS Protected Objects that are on different distribution channels</li> <li>2. Protection Manager Support allow only authorized operations on the object</li> <li>3. Objects are accessed on different channels and each AXCS stores its Action-Logs</li> <li>4. Via the AXCS sync general information on objects or information that allow SuperAXCS to recover Action-Logs from the different AXCSs are transferred to the SuperAXCS Collector</li> <li>5. SuperAXCS collects information</li> <li>6. Raw logs are extracted from SuperAXCS by CAMART</li> <li>7. Administrative reports are created</li> <li>8. Administrative Information Integrator format logs according to the predefined format for the user and transfers Action-Logs on CMS</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None

<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

**13.4.3 Distributor asks for administrative information**



<b>UCId</b>	UC13.4.3
<b>Use case</b>	Distributor asks for administrative information
<b>Description</b>	In response to the distributor request, the administrative information are transferred to the CMS of distributor
<b>Actors</b>	Distributor
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A Distributor wants to recover information on actions performed on the objects he has rights.</li> <li>2 Core accounting manager and reporting tool query the reporting webservice for obtaining the Action-Logs</li> <li>3 AXMEDIS Statistic or reporting tools query AXCS</li> <li>4 AXCS extracts the required Action-Logs and communicate them to the reporting tool</li> <li>5 Accounting report is generated.</li> <li>6 Accounting report is passed to the Administrative Information Integrator</li> <li>7 Data are loaded in the Distributor CMS</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None



### 13.4.4 Administrative information retrieval for distributors

<b>UCId</b>	UC13.4.4
<b>Use case</b>	Administrative information retrieval for distributors
<b>Description</b>	A content distributor requests administrative information about its own AXMEDIS objects
<b>Actors</b>	Content distributor
<b>Assumptions</b>	Content provider is registered and has distributed his/her AXMEDIS objects, users have used objects
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A content distributor requests administrative information about its own AXMEDIS objects.</li> <li>2 the administrative information integrator asks the AXCS to verify the content provider</li> <li>3 the AXCS certifies the content provider</li> <li>4 the administrative information integrator collects all Action-Logs related to the distributor objects</li> <li>5 the administrative information integrator records all collected data into the distributor CMS</li> </ol>
<b>Post-conditions</b>	Administrative information is inside the content provider database
<b>Variations</b>	<ol style="list-style-type: none"> <li>3a. content provider is not registered:                         <ol style="list-style-type: none"> <li>3a.1.the AXCS doesn't validate the content provider</li> <li>3a.2.the AXCS blocks the operation in progress</li> </ol> </li> </ol>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.4.5 Administrative information retrieval for collecting societies

<b>UCId</b>	UC13.4.5
<b>Use case</b>	Administrative information retrieval for collecting
<b>Description</b>	A collecting society requests administrative information about the objects under its territorial responsibility
<b>Actors</b>	Collecting society
<b>Assumptions</b>	Collecting society is registered and has to collect rights for objects distributed by different distributors on different channels
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A collecting society requests administrative information about AXMEDIS objects under its authority.</li> <li>2 The administrative information integrator asks the AXCS to verify the collecting society</li> <li>3 the AXCS certifies the collecting society</li> <li>4 The SuperAXCS collects information from the different AXCS that are on the different channels in order to collect Actions-Logs performed on objects for which the collecting society is authorized to collect rights</li> <li>5 the administrative information integrator collects from the SuperAXCS all Action-Logs related to the objects</li> <li>6 the administrative information integrator records all collected data into the collecting society CMS</li> </ol>
<b>Post-conditions</b>	Administrative information is inside the content provider database
<b>Variations</b>	<ol style="list-style-type: none"> <li>4a. content provider is not registered: <ol style="list-style-type: none"> <li>4a.1.the AXCS doesn't validate the content provider</li> <li>4a.2.the AXCS blocks the operation in progress</li> </ol> </li> </ol>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 13.5 Protection Manager Support/Server general

### 13.5.1 Protection Manager Support / Server

#### 13.5.1.1 Consumption of a protected and governed AXMEDIS object in a connected environment

<b>UCId</b>	UC13.5.1.1
<b>Use case</b>	Consumption of a protected and governed AXMEDIS object in a connected environment
<b>Description</b>	A user wants to consume a protected and governed AXMEDIS object
<b>Actors</b>	<p>End-user  AXMEDIS Tool (AXOM, Protection Processor, PMS Client)  PMS Server (Authorisation Server, License Database)  AXCS</p>
<b>Assumptions</b>	<p>User is registered.  The used AXMEDIS Tool is certified.  The user is connected.</p>

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user tries to consume a protected and governed AXMEDIS object using a certified AXMEDIS Tool.</li> <li>2 The Tool requests the clear-text content to the AXMEDIS Object Manager (AXOM).</li> <li>3 The AXOM (through the Protection Processor) performs a local Tool-integrity verification.</li> <li>4 The AXOM (through the Protection Processor) requests the Tool-integrity verification to the AXMEDIS Certifier and Verifier (AXCS), see Use Case “Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption”.</li> <li>5 As the AXMEDIS object is governed, the AXOM requests the authorisation to perform the action requested by the User to the PMS Client.</li> <li>6 The PMS Client obtains the status context information from Content Consumption status manager (Secure Cache).</li> <li>7 The PMS request the authorisation to the authorisation server. It sends an authorisation request that includes the user identification, the right, the resource, optionally the license(s) or its(their) identifier(s), the status information, the Tool identifier, (eventually) the Domain identifier.</li> <li>8 The authorisation server obtains the licenses associated to the user from the database of DRM licenses, if necessary, and performs the authorisation.</li> <li>9 If the result of the authorisation is negative: <ol style="list-style-type: none"> <li>9.1 the Authorization Server returns the reasons why not</li> <li>9.2 the user cannot consume the AXMEDIS object.</li> </ol> </li> <li>10 If the result of the authorisation is positive: <ol style="list-style-type: none"> <li>10.1 AXCS returns the Protection Info to PMS Client over a secure channel</li> <li>10.2 PMS Client temporarily stores the Protection Info in the local cache</li> <li>10.3 The Protection Processor uses the Protection Information to unprotect the content</li> <li>10.4 If the Protection Information are not cacheable, the Protection Information erases it from the local cache.</li> <li>10.5 The AXOM returns the clear-text content to the Tool</li> <li>10.6 The Tool can consume the content as it has been requested by the User</li> </ol> </li> </ol>
<b>Post-conditions</b>	The consumption of the AXMEDIS Objects generated an Action Log which reports all the needed data to track the event.
<b>Variations</b>	<p>If the local Tool-integrity verification fails, the Protection Processor disables the Tool and erase all sensible data from the Secure Cache.</p> <p>See “Verification of AXMEDIS users using AXMEDIS tools on a Device before content consumption” Use Case what happen when the verification of the tool integrity fails</p>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.1.2 Consumption of a protected and governed AXMEDIS object in a unconnected environment

<b>UCId</b>	UC13.5.1.2
<b>Use case</b>	Consumption of a protected and governed AXMEDIS object in a unconnected environment
<b>Description</b>	A user wants to consume a protected and governed AXMEDIS object

<b>Actors</b>	End-user AXMEDIS Tool (AXOM, Protection Processor, PMS Client, License Interpreter)
<b>Assumptions</b>	User is registered. The AXMEDIS Tool is certified. The Tool does not have connection. The AXMEDIS object has been downloaded on the Device. The licenses and the Protection Information related to the User and the AXMEDIS Object are cacheable and they have been previously cached in the Tool
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user tries to consume a protected and governed AXMEDIS Object.</li> <li>2 The Tool requests the clear-text content to the AXMEDIS Object Manager (AXOM).</li> <li>3 The AXOM (through the Protection Processor) performs a local Tool-integrity verification.</li> <li>4 As the AXMEDIS object is governed, the AXOM requests the authorisation to perform the action requested by the User to the PMS Client.</li> <li>5 The PMS Client obtains the status context information from Content Consumption status manager (Secure Cache).</li> <li>6 The PMS Client (through the License Interpreter) performs the authorisation verification algorithm on the basis of: the user identification, the right, the resource, the license(s), the context information, the Tool identifier and (eventually) the Domain identifier.</li> <li>7 If the result of the authorisation is negative, it returns the reasons why not and the user cannot consume the AXMEDIS object.</li> <li>8 If the result of the authorisation is negative: <ol style="list-style-type: none"> <li>8.1 the PMS Client returns the reasons why not</li> <li>8.2 the user cannot consume the AXMEDIS object.</li> </ol> </li> <li>9 If the result of the authorisation is positive: <ol style="list-style-type: none"> <li>9.1 The Protection Processor uses the Protection Information (previously cached in the Secure Cache) to unprotect the content</li> <li>9.2 The AXOM returns the clear-text content to the Tool</li> <li>9.3 The Tool can consume the content as it has been requested by the User</li> </ol> </li> </ol>
<b>Post-conditions</b>	The consumption of the AXMEDIS Objects generated an Action Log which reports all the needed data to track the event. The Action Log has been stored in the Secure Cache. The whole list of cached Action Logs will be sent to the AXCS at the next connected consumption. See Use Case “Consumption of a protected and governed AXMEDIS object in a unconnected environment”.
<b>Variations</b>	If the local Tool-integrity verification fails, the Protection Processor disables the Tool and erase all sensible data from the Secure Cache.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.1.3 Protection of an AXMEDIS object

<b>UCId</b>	UC13.5.1.3
<b>Use case</b>	Protection of an AXMEDIS object
<b>Description</b>	A user wants to protect an AXMEDIS Object
<b>Actors</b>	AXMEDIS Object Creator, Protection tool engine
<b>Assumptions</b>	The user has logged in and his identity has been validated. AXMEDIS Editor is running and an AXMEDIS object is opened in the editor

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A User wants to protect an AXMEDIS object.</li> <li>2 The User uses the Protection Editor and Viewer (within the AXMEDIS Editor) in order to create the Protection Info for the given AXMEDIS Object.</li> <li>3 When the User want to save the AXMEDIS Object, Protection Processor protects it on the basis of the provided Protection Info and store them within the AXMEDIS Object.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Once the AXMEDIS Object will be registered, the Protection Info will be removed from the object itself and it will be stored in the AXCS. For more details, see Use Cases “Storage/update of protection information of an AXMEDIS Object to the AXCS” and “Registration of metadata about a new object”.

#### 13.5.1.4 Protection and association of licenses of/to an AXMEDIS object

<b>UCId</b>	UC13.5.1.4
<b>Use case</b>	Protection and association of licenses of/to an AXMEDIS object
<b>Description</b>	A user wants to protect an AXMEDIS object and to associate to it license(s).
<b>Actors</b>	AXMEDIS Object creator, Protection Tool engine
<b>Assumptions</b>	The user has logged in and his identity has been validated. AXMEDIS Editor is running and an AXMEDIS object is opened at the editor
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user wants to protect an AXMEDIS object and to associate to the AXMEDIS object a set of licenses.</li> <li>2 AXMEDIS editor makes use of Encryption support (see use case Encryption)</li> <li>3 If the encryption is correct, and the encryption information has been generated to protect this object, it has to be stored (see use case Storage of security information)</li> <li>4 PMS generates the appropriate rights expressions that contain: <ol style="list-style-type: none"> <li>4.1 The license(s) that the user wants to associate to this AXMEDIS object. The licenses can be associated including them within the AXMEDIS object, or references, or a license service referenced within the AXMEDIS object.</li> </ol> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 13.5.1.5 Renewal of IPMP information after the detection of a succeed attack (connected)

<b>UCId</b>	UC13.5.1.5
<b>Use case</b>	Renewal of IPMP information after the detection of a succeed attack
<b>Description</b>	The protection of an AXMEDIS object has been violated, and a renewal of it is needed
<b>Actors</b>	AXCS, PMS
<b>Assumptions</b>	

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A succeed attack over the protection of an AXMEDIS object has been detected by AXCS (or external detection).</li> <li>2 New key for protecting the object is generated</li> <li>3 The AXMEDIS object is re-protected with the new key (new algorithm)</li> <li>4 The AXMEDIS object, together with its protection information are stored in the database. It is also indicated that the protection method has changed in order to inform the users accessing the protected AXMEDIS object</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 13.5.2 DRM Support

### 13.5.2.1 License creation for new content

<b>UCId</b>	UC13.5.2.1
<b>Use case</b>	License creation for new content.
<b>Description</b>	An actor requests a license to consume or distribute content.
<b>Actors</b>	Content creator, Distributor, Integrator
<b>Assumptions</b>	AXMEDIS object exists and is uniquely identified
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor requests a license.</li> <li>2 The License Generator obtains the relevant parameters to generate the license (principal, right/s, resource and conditions).</li> <li>3 The License Generator creates the license</li> <li>4 The License Verifier validates the license</li> <li>5 If the license is valid, the License Manager stores the license in the database of DRM licenses. If not, returns an alert with an explicative message.</li> <li>6 The License Generator returns to the actor the license ID or the license in clear-text or both.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Implement it as a web service interface.
<b>Issues</b>	None

### 13.5.2.2 License creation for cross-media content

<b>UCId</b>	UC13.5.2.2
<b>Use case</b>	License creation for cross-media content.
<b>Description</b>	An actor requests a license to consume, create or distribute cross-media content.
<b>Actors</b>	Content Creator, Distributor, Integrator
<b>Assumptions</b>	AXMEDIS objects that have to be integrated exist, are uniquely identified and have some licenses associated

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor wants to create a license for cross-media content from several existing AXMEDIS objects</li> <li>2 The License Generator obtains from the License Manager all the licenses associated to the original AXMEDIS objects</li> <li>3 The License Generator derives a new license from the obtained licenses</li> <li>4 The License Verifier validates the new license</li> <li>5 The License Verifier verifies that the derived conditions are consistent</li> <li>6 If the license is valid and the conditions are consistent the License Manager stores the license in the database of DRM licenses. If not, returns an alert with an explicative message.</li> <li>7 The License Generator returns to the actor the license ID or the license in clear-text or both.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Implement it as a web service interface.
<b>Issues</b>	None

### 13.5.2.3 License verification against parent licenses

<b>UCId</b>	UC13.5.2.3
<b>Use case</b>	License verification against parent licenses
<b>Description</b>	A newly created license if verified against the chain of parent licenses
<b>Actors</b>	License creator
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user creates a new license.</li> <li>2 User asks for verification of the newly created license. The license is checked against the chain of parent licenses (at least the direct parent). To do so, the license is checked using a simplified version of the MPEG-21 authorisation algorithm. First, we look for the direct parent license and check that conditions are exactly the same. If not, we iterate until there are no more parent licenses for that license or the condition is satisfied.</li> <li>3 If the license is verified, then it is stored.</li> <li>4 If not, the license is not verified and the user is informed.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.2.4 License verification against PAR

<b>UCId</b>	UC13.5.2.4
<b>Use case</b>	License verification against PAR
<b>Description</b>	A newly created license if verified against the object PAR

<b>Actors</b>	License creator
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user creates a new license.</li> <li>2 User asks for verification of the newly created license. The license is checked against object PAR. To do so, the license is checked using a simplified version of the MPEG-21 authorisation algorithm.</li> <li>3 If the license is verified, then it is stored.</li> <li>4 If not, the license is not verified and the user is informed.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.2.5 User authorisation on unconnected environment

<b>UCId</b>	UC13.5.2.5
<b>Use case</b>	User authorisation based on unconnected environment
<b>Description</b>	When a user wants to perform an action over a resource, the authorisation server checks if the user has the appropriate permissions according to the license terms.
<b>Actors</b>	Distributors, end-users.
<b>Assumptions</b>	Licenses are expressed in MPEG-21 REL.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The PMS server is offline and you want to perform an action.</li> <li>2 The PMS Client receives an authorisation request that includes an ActionLog with the user identification, the right, the resource and optionally the license(s) or its(their) identifier(s).</li> <li>3 If the object is protected, the PMS Client request to the SecureCache the Protection Information. (If it's not found, the PMS Client returns an error and stops).</li> <li>4 Locally the authorisation support obtains the licenses associated to the user from the secure cache of DRM licenses.</li> <li>5 The authorisation support performs the authorisation.</li> <li>6 If the result of the authorisation is positive the user is authorised. If not, it returns the reasons why not.</li> <li>7 A complete ActionLog is stored in the secure cache.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Use of ODRL licenses.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Implement it as a web service interface.
<b>Issues</b>	None

### 13.5.2.6 User authorisation on semiconnected environment (PMS server Online, AXCS offline)

<b>UCId</b>	UC13.5.2.6
<b>Use case</b>	User authorisation on semi-connected environment
<b>Description</b>	When a user wants to perform an action over a resource, the authorisation server checks if the user has the appropriate permissions according to the license terms.
<b>Actors</b>	Distributors, end-users.
<b>Assumptions</b>	Licenses are expressed in MPEG-21 REL.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The PMS server is online, but AXCS is offline and you want to perform an action.</li> <li>2 The PMS Client receives an authorisation request that includes an ActionLog with the user identification, the right, the resource and optionally the license(s) or its(their) identifier(s).</li> <li>3 If the object is protected, the PMS Client requests to the SecureCache the Protection Information. (If it's not found, the PMS Client returns an error and stops).</li> <li>4 The PMS Client sends the request to the PMS Server.</li> <li>5 In the PMS Server, the authorisation support obtains the licenses associated to the user from the secure cache of DRM licenses.</li> <li>6 The authorisation support performs the authorisation.</li> <li>7 If the result of the authorisation is positive the user is authorised. If not, it returns the reasons why not.</li> <li>8 A complete ActionLog is stored in the secure cache (locally).</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Use of ODRL licenses.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Implement it as a web service interface.
<b>Issues</b>	None

### 13.5.2.7 User authorisation on fully connected environment (PMS server Online, AXCS online)

<b>UCId</b>	UC13.5.2.7
<b>Use case</b>	User authorisation on fully connected environment
<b>Description</b>	When a user wants to perform an action over a resource, the authorisation server checks if the user has the appropriate permissions according to the license terms.
<b>Actors</b>	Distributors, end-users.
<b>Assumptions</b>	Licenses are expressed in MPEG-21 REL.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The PMS server is online, and AXCS is online and you want to perform an action.</li> <li>2 The PMS Client receives an authorisation request that includes an ActionLog with the user identification, the right, the resource and optionally the license(s) or its(their) identifier(s).</li> <li>3 The PMS Client sends the request to the PMS Server.</li> <li>4 In the PMS Server, the authorisation support obtains the licenses associated to the user from the secure cache of DRM licenses.</li> <li>5 If the object is protected, the PMS Server request to the AXCS Server the Protection Information. (If it's not found, the PMS Server returns an error and stops).</li> <li>6 The authorisation support performs the authorisation.</li> <li>7 If the result of the authorisation is positive the user is authorised. If not, it returns the reasons why not.</li> <li>8 A complete ActionLog is stored in the AXCS.</li> <li>9 The ActionLog is returned, with the authorisation result, and the Protection Information to the PMS Client.</li> <li>10 The ActionsLog, is also stored in the SecureCache.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	Use of ODRL licenses.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Implement it as a web service interface.

<b>Issues</b>	None
---------------	------

### 13.5.2.8 Navigation of licensing information

<b>UCId</b>	UC13.5.2.8
<b>Use case</b>	Navigation on the licensing information
<b>Description</b>	A user wants to view graphically the information related to his licenses.
<b>Actors</b>	Content creators, distributors and end-users.
<b>Assumptions</b>	None
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user requests to see graphically the information within his licenses.</li> <li>2 The list of licenses associated to this user is shown.</li> <li>3 The user chooses one of his licenses.</li> <li>4 The license is shown graphically</li> <li>5 The user navigates on the information of the license.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.2.9 Rights Expression Translator

<b>UCId</b>	UC12.5.2.9
<b>Use case</b>	Rights Expression Translator
<b>Description</b>	The system wants to translate a valid license (for instance, a mobile profile) from a REL into another, providing interoperability.
<b>Actors</b>	System
<b>Assumptions</b>	A valid license exists
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when the system detects that a REL translation is needed.</li> <li>2 Selection of a supported REL destination from a list</li> <li>3 Translate the license</li> <li>4 Check if the destination license is valid or not</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	If the destination license is not valid, the translation will not be possible and the system will show a message informing of this.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.2.10 License migration

<b>UCId</b>	UC12.5.2.3
<b>Use case</b>	License migration among different PMSs
<b>Description</b>	A user requests authorization to consume a content to a PMS not owning the pertinent license (this can happen, for instance, when the user move to another geographic area)
<b>Actors</b>	Users, PMSs.
<b>Assumptions</b>	The requested license is present in almost one reachable PMS
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A User asks for an authorization (for object consumption) to the usual PMS</li> <li>2 The PMS checks if it locally owns the license related with the client request</li> <li>3 If not, it asks for the needed license to the connected (known) PMSs</li> <li>4 The PMS that retains the license deliver the license to the requesting PMS</li> <li>5 The PMS which is serving the user request, stores the acquired license in its own storage and checks if the requested consumption matches with the acquired license data</li> </ol>
<b>Post-conditions</b>	The PMS knowledge can grow
<b>Variations</b>	Storage of acquired license can be placed or not
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 13.5.2.11 Cooperative Authorization Check

<b>UCId</b>	UC12.5.2.3
<b>Use case</b>	Cooperative authorization check among different PMSs
<b>Description</b>	A user requests authorization to consume a content to a PMS not owning the pertinent license (this can happen, for instance, when the user move to another geographic area)
<b>Actors</b>	Users, PMSs.
<b>Assumptions</b>	The requested license is present in almost one reachable PMS
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A User asks for an authorization (for object consumption) to the usual PMS</li> <li>2 The PMS checks if it locally owns the license related with the client request</li> <li>3 If not, it asks for performing a match between licenses and requested consumption to the connected (known) PMSs</li> <li>4 The PMS that retains the license perform locally the requested check and answer to the requesting PMS with result</li> <li>5 The PMS which is serving the user request, answers to the requesting user</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 13.6 Encryption/Decryption Support

### 13.6.1 Encryption

<b>UCId</b>	UC13.6.1
<b>Use case</b>	Encryption of AXMEDIS object
<b>Description</b>	An actor wants to save a protected AXMEDIS object
<b>Actors</b>	Any that can save an AXMEDIS object
<b>Assumptions</b>	AXMEDIS Editor is running and an AXMEDIS object is opened at the editor

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor wants to protect an AXMEDIS object</li> <li>2 Check if actor has permission to protect the object</li> <li>3 If actor has permission, the key for encrypting the object is recovered from its storage (license, etc)</li> <li>4 The object is encrypted and can be saved in a protected manner</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	<p>The key for encrypting the object does not exist and it has to be created by calling AXMEDIS certifier</p> <p>The Protection Engine wants to protect some objects</p>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Library providing symmetric encryption functionalities
<b>Issues</b>	None

### 13.6.2 Decryption

<b>UCId</b>	UC13.6.2
<b>Use case</b>	Decryption of AXMEDIS object
<b>Description</b>	An actor wants to open a protected AXMEDIS object
<b>Actors</b>	Any that can open an AXMEDIS object
<b>Assumptions</b>	AXMEDIS Editor or AXMEDIS Viewer is running or the actor double clicks on an AXMEDIS object to open it (Windows Systems)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The use case begins when an actor calls the “Open object” button on AXMEDIS Editor or AXMEDIS Viewer</li> <li>2 Check if actor has permission to open the object</li> <li>3 If actor has permission, the key for decrypting the object is recovered</li> <li>4 The object is decrypted and AXMEDIS Editor or Viewer can show it</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Library providing symmetric decryption functionalities
<b>Issues</b>	None

### 13.6.3 Encryption of symmetric key

<b>UCId</b>	UC13.6.3
<b>Use case</b>	Encryption of AXMEDIS object symmetric key using public key techniques
<b>Description</b>	A symmetric key for an AXMEDIS object is encrypted with asymmetric encrypting techniques for secure storage
<b>Actors</b>	The creator of the AXMEDIS object whose symmetric key is going to be encrypted
<b>Assumptions</b>	<p>A symmetric key for an AXMEDIS object has been generated</p> <p>A pair of public key – private key have been generated for the creator actor</p>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Symmetric key for AXMEDIS object is encrypted with the public component of the creator’s asymmetric key</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Library providing asymmetric key encryption
<b>Issues</b>	None

### 13.6.4 Decryption of symmetric key

<b>UCId</b>	UC13.6.4
<b>Use case</b>	Decryption of AXMEDIS object symmetric key using public key techniques
<b>Description</b>	A symmetric key for an AXMEDIS object is decrypted using asymmetric encrypting techniques to allow AXMEDIS object decryption

<b>Actors</b>	The actor that wants to use protected AXMEDIS object
<b>Assumptions</b>	The actor that wants to use the protected AXMEDIS object has received the symmetric key for decrypting the object. This symmetric key is encrypted with the public component of the actor's asymmetric key. Actor has got the corresponding private component to decrypt the symmetric key and then be able to open AXMEDIS object
<b>Steps</b>	1 Symmetric key for AXMEDIS object is decrypted with the private component of the actor's asymmetric key
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Library providing asymmetric key encryption
<b>Issues</b>	None

## 14 AXMEDIS Player

As mainly a subset in functionality of an AXMEDIS Editor, and strongly interacting with distribution to end users, the AXMEDIS Player shares some of the most common use cases that are also identified for the Editor and common distribution scenarios. These use cases apply when a Player is considered instead of an Editor. These use cases are:

- Acquisition of AXMEDIS objects
- Viewing/Using of AXMEDIS objects
- Load and save AXMEDIS objects
- Invoking an internal viewer (/editor)
- Managing a digital resource by respecting the DRM in an Internal Viewer/Editor
- Closing an internal viewer (/editor)
- Invoking an external tool with a digital resource belonging to the AXMEDIS object
- Transferring a digital resource to an external tool
- Decryption
- Decryption of symmetric key
- User Software Installation
- User Registration

Among the above listed use cases, some apply only to a PC platform, or platform with considerable resources and functionality: 4.3.1, 4.3.5, 14.2.1, 14.2.2

The other use cases apply more in general to different Players, including portable devices, by considering suitable adaptations (for instance a suitable pointing/clicking device may be available instead of the mouse).

### 14.1 AXMEDIS Player on PC, Tablet PC

This section contains additional use cases which are more specific of a Player tool. While “Local adaptation of Content in Internal Players/Viewers” and “Annotate for personal use” may be more suitable for devices with considerable resources and more sophisticated interfaces like PCs, the other use cases apply as well to less powerful portable devices like PDAs or mobile Players

#### 14.1.1 Content Recording for Playtime Shift

<b>UCId</b>	UC14.1.1
<b>Use case</b>	Content Recording for Playtime Shift
<b>Description</b>	The user can store some content in a backup support to possibly play content with a time shift from the moment when it was downloaded.
<b>Actors</b>	The Content Consumer (user)

<b>Assumptions</b>	The user has a function in the terminal that allows the operation of backup (or record; the function must ensure the integrity of the copied AXMEDIS Object, taking into account that an Object could be internally combined with other Objects). The Backup Function has to be expressly authorized in the license terms.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user select from a content distributor catalogue an AXMEDIS Object to download</li> <li>2 The client terminal, if license terms for the Object allow this, activate the Backup Function</li> <li>3 The user specifies the “title” with which the AXMEDIS Content has to be recorded.</li> <li>4 The user execute the Backup/Record Function</li> <li>5 At a later time, the Player can be started in playback mode to play a selected recorded “title”.</li> </ol>
<b>Post-conditions</b>	None.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	The AXMEDIS Content can be deleted during the playback operation, according to the license terms.
<b>Design suggestions</b>	The Player should have a playback function allowing access to stored “titles” without an explicit path access for the user.
<b>Issues</b>	None.

#### 14.1.2 Fast-forward of Content in Internal Players/Viewers

<b>UCId</b>	UC14.1.2
<b>Use case</b>	Fast-forward of Content in Internal Players/Viewers
<b>Description</b>	The User wants to play a digital resource faster for a quick preview or for fast access to a later sequence
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• AXMEDIS Player is open</li> <li>• An object is opened within the AXMEDIS Player</li> </ul>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The User select the Play command</li> <li>2 The system activates the proper internal player/viewer.</li> <li>3 The User select the fast-forward command</li> <li>4 The activated viewer/player inside the AXMEDIS Player starts skipping frames at appropriate rate to speed-up playback speed.</li> <li>5 When the User releases the fast-forward command, the viewer/player returns to normal playback mode.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	End User usually can only “play” an AXMEDIS object, so only internal “players/viewers” should be invoked, not editors.

#### 14.1.3 Local adaptation of Content in Internal Players/Viewers

<b>UCId</b>	UC14.1.3
<b>Use case</b>	Local adaptation of Content in Internal Players/Viewers
<b>Description</b>	The User wants to play more digital resources possibly requiring a resource management in real-time
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	<ul style="list-style-type: none"> <li>• AXMEDIS Player is open</li> <li>• One or more objects are opened by the user within the AXMEDIS Player</li> </ul>

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The User select the Play command</li> <li>2 The system activates the proper internal player/viewer.</li> <li>3 The User select again the Play command for a second Object</li> <li>4 The activated viewers/players inside the AXMEDIS Player monitor the resource availability: they possibly start skipping frames at appropriate rate to maintain system stability.</li> <li>5 When the User stops one of the object playbacks, the viewer/player returns to normal playback mode.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	End User usually can only “play” an AXMEDIS object, so only internal “players/viewers” should be invoked, not editors.

#### 14.1.4 Annotate for personal use

<b>UCId</b>	UC14.1.4
<b>Use case</b>	Annotate for personal use
<b>Description</b>	<p>The end user is able to add personal annotations to content by using the Player. The annotations can be text, graphics, recorded voice etc. that refer to content. The annotations do not modify the original content. The annotations are stored locally and separated from the content. The Player has the capacity to associate the annotations with the content so that the user knows to which content the annotations belong. The user can play the content and take the notes at the same time. When the content is a moving picture or a song the Player can add time stamps to the annotations.</p> <p>The Player can also handle the use of spatial stamps to accompany the user annotations. A spatial stamp could be used to associate an annotation to a certain position of a still picture.</p>
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The end user starts playing content.</li> <li>2 The end user exploits the Player reduced editing functions to add some annotation related to the played content.</li> <li>3 The player takes care of associating the annotations with the content without modifying the content itself. The Player records the time and space coordinates of the annotation respect to the content.</li> <li>4 The final user finishes playing and annotating the content.</li> <li>5 Annotations are stored independently from AXMEDIS Objects containing the content but keeping a reference to them.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	This can be realized by adding a limited part of the editor functionality and allowing to save notes in a different way
<b>Issues</b>	None

#### 14.1.5 Local User Profiles

<b>UCId</b>	UC13.1.5
<b>Use case</b>	Local User Profiles

<b>Description</b>	The Player is able to handle local user profiles. Different users may use the same Player configured in a slightly different manner so that the Player behaves or looks different. The profile can specify general settings of the player for instance: language of the user interface, default volume, default decoder if several of them are available or default skin. The users are able to configure their own profile.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The end user configures and saves a user profile.</li> <li>2 Every time the Player is opened it checks the user profile and adapts itself to the user preferences.</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	A privileged user could set the profile for one or more non-privileged users. For instance the user Parent could define the Child profile. The Child profile could state that e.g. no content rated R (Restricted in American movies) can be played or that no content can be played after a given time.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 14.1.6 History of the last played contents

<b>UCId</b>	UC13.1.6
<b>Use case</b>	User profiles
<b>Description</b>	The Player is able to record the name and location of last played contents. The list of the last played contents is available to the end user. The list is stored locally in the user platform. The length of the list is defined by the end user. The end user can disable this functionality.
<b>Actors</b>	End User
<b>Assumptions</b>	
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Every time the user plays content the Player keeps track of it in a local list.</li> <li>2 The user can know what the last played contents are.</li> </ol>
<b>Post-conditions</b>	none
<b>Variations</b>	none
<b>Asynchronous actions</b>	none
<b>Design suggestions</b>	
<b>Issues</b>	

## 15 AXMEDIS for Distribution via Internet

### 15.1 Back Office Management

#### 15.1.1 Creating a New Mediaclub

<b>UCId</b>	UC15.1.1
<b>Use case</b>	Creating a new Mediaclub setup
<b>Description</b>	Set up a new mediaclub in the cms
<b>Actors</b>	System Manager (sys mng)
<b>Assumptions</b>	The system is up and running and fully configured; Actors have network access to the management interface (web). All technical info needed to configure the mediaclub are provided by the Content distributor
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 (sys mng) log in to the system and add a new project (name and description)</li> <li>2 (sys mng) configure the mediaclub website publishing targets and publishing modes (static pages, dynamic, etc)</li> <li>3 (sys mns) create the projects content repository witch will contains the contents types definition and all contents that will be included in the project</li> <li>4 (sys mng) creat the project media repository witch contains binaries content as images, video stream, audio stream, etc</li> <li>5 (sys mng) Define feed import rules</li> <li>6 (sys mng) Define referred publishing rules, if needed</li> <li>7 (sys mng) configure the project administrator</li> <li>8 (sys mng) Save configuration</li> </ol>
<b>Post-conditions</b>	Test page should be displayed in the website publishing targets and prj mng successfully log in
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

#### 15.1.2 Mediaclub Setup

<b>UCId</b>	UC15.1.2
<b>Use case</b>	Mediaclub set up
<b>Description</b>	Define all mediaclub feactures in the cms
<b>Actors</b>	Project Manager (prj mng)
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 (prj mng) log in to the system and load the project settings form (name and description)</li> <li>2 (prj mng) configure the mediaclub website sections</li> <li>3 (prj mng) create the projects content types (xsl schema; xsl target and taget layout)</li> <li>4 (prj mng) create content categories and media categories three</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.3 Medioclub Accounts and Permission Management

<b>UCId</b>	UC15.1.3
<b>Use case</b>	Medioclub accounts and permissions
<b>Description</b>	Manage a medioclub accounts and their permissions
<b>Actors</b>	Project Manager (prj mng)
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 (prj mng) log in to the system and load the project account management form</li> <li>2 (prj mng) Create a new project account defining personal details, user id, password</li> <li>3 (prj mng) Define account permission (Editor, publish authorizer, project manager .....</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.4 Medioclub Project Uploading and publishing contents

<b>UCId</b>	UC15.1.4
<b>Use case</b>	Medioclub publishing
<b>Description</b>	Upload contents in the cms and publish them in the related medioclub site
<b>Actors</b>	(editor) actors allowed to put contents in the medioclub, (publisher) actors allowed to authorize content publishing
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 (editor) log in to the system and loads the select new content action</li> <li>2 (editor) choose the content type and define a content name</li> <li>3 (editor) fill all fields required from the defined content type</li> <li>4 (editor) save content and choose one or more publishing targets</li> <li>5 (editor) submit content to authorization for publishing</li> <li>6 (publisher) authorize or reject the publish request</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.5 Medioclub Project Acquiring AXMEDIS content

<b>UCId</b>	UC15.1.5
<b>Use case</b>	Medioclub and AXMEDIS content
<b>Description</b>	Set up a new medioclub in the cms
<b>Actors</b>	Project Manager (prj mng)
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)

<b>Steps</b>	1 (prj mng) search a specific content on a AXMEDIS p2p network 2 (prj mng) select AXMEDIS content and view all meta data infos 3 (prj mng) acquire license (if needed) and refer the object in the medioclub contents
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.6 Medioclub Project define payment gateway entry

<b>UCId</b>	UC15.1.6
<b>Use case</b>	Medioclub payments system setup
<b>Description</b>	Enable the payment gateway to provide payment service to the specific medioclub
<b>Actors</b>	System Manager (sys mng)
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)
<b>Steps</b>	1 (sys mng) log in to the system and go in to payment management section 2 (sys mng) configure a new medioclub shop in the payment gateway giving (name, description, other details) 3 (sys mng) Define payment methods available for the medioclub 4 (sys mng) configure the shop administrator 5 (sys mng) Save configuration
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.7 Medioclub Shop payment Management

<b>UCId</b>	UC15.1.7
<b>Use case</b>	Medioclub shop payments configuration
<b>Description</b>	Configure a medioclub shop in the payment gateway
<b>Actors</b>	Shop Manager (shop mng)
<b>Assumptions</b>	The system is up and running and fully configured; actors have network access to the management interface (web)
<b>Steps</b>	1 (shop mng) log in to the system and go in to payment management section 2 (sys mng) configure medioclub call back URL for success, failure and error transaction 3 (shop mng) Choose payment methods available for the medioclub 4 (sys mng) upload schema and graphical components needed to build the payments transaction pages that will be shown to the end user
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 15.1.8 Mediaclub Shop Management refund a transaction

<b>UCId</b>	UC15.1.8
<b>Use case</b>	Mediaclub refund management
<b>Description</b>	refund a payment transaction in a mediacub shop
<b>Actors</b>	Shop Manager (shop mng)
<b>Assumptions</b>	Customer have provided transaction details and is proven that he hasn't had the digital goods
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 (shop mng) search the transaction id and or the user id in the transaction list</li> <li>2 (shop mng) load the transaction details and check if everithing is ok</li> <li>3 (shop mng) starts transaction refund process</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 15.2 End User Client configuration

### 15.2.1 User Software Installation

<b>UCId</b>	UC15.2.1
<b>Use case</b>	User Software Installation
<b>Description</b>	The user installs the AXMEDIS Client Application
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	<p>User has PC or STB based on most common Operating Systems: Windows XP, 2000, ME; MacOS X; Linux.</p> <p>User has an Internet connection (DSL preferably)</p>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user obtains the AXMEDIS Client Application Setup (e.g., from the Internet, from a CD, ...)</li> <li>2 The user runs the AXMEDIS Client Application Setup</li> <li>3 The user follows the steps of the installation</li> <li>4 Some information entered by the user is used to create his profile, that is securely communicated via Internet to MediaClub installation</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	The exact form of the Setup, and the installation steps, could depend on the exact operating system of the user's PC, and on its configuration. Also, the user profile (country, language, gender, ...) may influence some steps (e.g., subscription to language-specific services).
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	<p>Different Setups applications could be distributed according to country, bundling with Tiscali DSL set-up packages, commercial promotions, etc.</p> <p>The Setups should contain minimal 'intelligence' and be driven by the client management server, in order to make updates easier.</p> <p>The client media Server shall implement algorithms to detect the user language and provide him useful information on the service.</p>
<b>Issues</b>	None.

### 15.2.2 User Registration

<b>UCId</b>	UC15.2.2
<b>Use case</b>	User Registration

<b>Description</b>	The user register himself in order to access the MediaClub service
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has successfully installed the software AXMEDIS components and has an email address
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user runs the MediaClub web service registration procedure</li> <li>2 The user enters or updates his personal profile, including user e-mail and marketing information, that is securely stored in the MediaClub</li> <li>3 The user obtains via email required authorizations (e.g., login/password) to access the MediaClub. (This could require paying a subscription fee, a pre-paid amount, etc.)</li> <li>4 The user replies via email to confirm registration</li> <li>5 The AXMEDIS Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)</li> </ol>
<b>Post-conditions</b>	The user is ready to use the MediaClub service and access the published Content catalogue. (Access can be restricted only to some components)
<b>Variations</b>	<p>The user may re-run the procedure to update his profile.</p> <p>In some situations this procedure could be automatic and hidden to the user.</p> <p>If the user already has a profile on the Server, his profile is restored in the local installation (e.g., user preferences, history, rights ...). This may occur, e.g., if the user is installing the Client Application on a different computer.</p>
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	user profile may be used in order to personalize content presentation, messages, etc.
<b>Issues</b>	None.

### 15.3 User login

#### 15.3.1 Authentication trough AXMEDIS client

<b>UCId</b>	UC15.3.1
<b>Use case</b>	User Login
<b>Description</b>	The user authenticate itself into the AXMEDIS system, via the AXMEDIS client (player/editor etc.)
<b>Actors</b>	Any kind of user
<b>Assumptions</b>	<p>User has a properly installer AXMEDIS client on his Device</p> <p>User is already registered in the AXMEDIS system</p>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user launch the AXMEDIS client software</li> <li>2. The software prompt the User for its login/password/certificate path or whatever is used to authenticate him</li> <li>3. The User authenticate itself</li> <li>4. The AXMEDIS Client application starts a Session for this user, granting the rights associated to his profile</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	Models where the player is certified for a given device only and can run unrestricted on this device ?
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 15.3.2 Authentication trough an external SSO system

<b>UCId</b>	UC15.3.2
<b>Use case</b>	User Login

<b>Description</b>	This Case take place in a system where the AXMEDIS client belongs to a wider software system which provides its own SSO authentication mechanism. This will typically be the case in a school, where a VLE (Virtual Learning Environment) uses such mechanism to control access to each one of its modules. In such a system a second authentication step into the AXMEDIS system is not desirable.
<b>Actors</b>	End users
<b>Assumptions</b>	The distributor of the VLE integrates the AXMEDIS player into his own software. The VLE distributor has deal an agreement with a Distributor to provide some resources to its users, and to automatically make the VLE users be registered AXMEDIS users
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. The user launch the VLE client software (?)</li> <li>2. The software prompt the User for its login/password/certificate path or whatever is used to authenticate him into the VLE</li> <li>3. The User authenticate itself</li> <li>4. The VLE checks authorisations and grant rights Accordingly. <ol style="list-style-type: none"> <li>4.1 if the User belongs to the Agreement with the AXMEDIS content provider, the VLE opens a session for him into the AXMEDIS system. AXMEDIS player grants rights according to the user's profile in its own registered users DB.</li> <li>4.2 If the User does not belong to the Agreement with AXMEDIS content provider, the VLE does not open a session for him into the AXEMDIS system. When the User tries to access AXMEDIS player, the player request authentication.</li> </ol> </li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	<p>The agreement between the VLE and content provider may take many forms :</p> <ol style="list-style-type: none"> <li>1. Agreement on a per user basis : e.g. each student user may be registered by itself into the AXMEDIS content provider system, thus allowing a fine grained tuning on which content is available for each one. This is the preferred solution</li> <li>2. Agreement on domain basis : e.g. the domain is a school and whoever logs in from this domain has access to the whole pre-agreed content.</li> </ol>
<b>Asynchronous actions</b>	The VLE maintains a DB of its user. This DB must be synchronized in some ways with the AXMEDIS registered user's DB (either a batch process or a live bridge between the two DBs)
<b>Design suggestions</b>	
<b>Issues</b>	<p>AXMEDIS DB of users must be able at least to import users from the VLE DB. Maybe a dynamic bridge between the two DB may be a better solution (that is, the AXMEDIS DB is able to interrogate another DB when not finding the user in its own DB).</p> <p>This disposal should not break any AXMEDIS security rule.</p>

## 15.4 Catalogue Browsing

### 15.4.1 Catalogue Listing

<b>UCId</b>	UC15.4.1
<b>Use case</b>	Catalogue Listing
<b>Description</b>	The user accesses a MediaClub web page containing the catalogue list of AXMEDIS content in order to select and playback content. He browses and previews the content listed in order to find the interesting content for him. Content may be delivered in unicast or multicast mode depending if content is on demand or live.
<b>Actors</b>	The Content Consumer (user)

<b>Assumptions</b>	The user shall have an active Internet Connection needed to reach the MediaClub web page where the proposed AXMEDIS Content Web List is published. The user shall know the URL where the MediaClub is published.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user reaches the AXMEDIS Catalogue List</li> <li>2 AXMEDIS content is displayed according to various criteria (type, author, content producer, production date)</li> <li>3 user selects and accesses content by clicking on the reference</li> </ol>
<b>Post-conditions</b>	User accesses Content Access or sub-catalogue List
<b>Variations</b>	The catalogue list is used also for listing sub-catalogue listings such as content categories or search results This AXMEDIS Content catalogue List could be published by third party distributor (e.g., OD2, iLabs, Sejer, etc.). XML data will enable lay-out flexibility on the third party distributor website.
<b>Asynchronous actions</b>	User must be opted with FAQ, HELP and customer care forms urls
<b>Design suggestions</b>	The user may search catalogue based on key words or free text search Possible previews related to the AXMEDIS Object may be provided
<b>Issues</b>	None.

#### 15.4.2 Catalogue Searching

<b>UCId</b>	UC15.4.2
<b>Use case</b>	Catalogue Searching
<b>Description</b>	The user searches content in the MediaClub or Content on P2P network
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user shall have an active Internet Connection needed to reach the MediaClub web page where the proposed AXMEDIS Content Web List is published. The user shall know the URL where the MediaClub is published.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user accesses a form within the AXMEDIS Client plug-in where he can operate keyword or free-text searches</li> <li>2 Search results of AXMEDIS content is displayed according to various criteria (type, author, content producer, production date) and whether content is available in MediaClub or on the P2P network</li> <li>3 user selects and accesses content by clicking on the reference</li> </ol>
<b>Post-conditions</b>	User accesses Content Access
<b>Variations</b>	This AXMEDIS catalogue Search could be published by third party distributor (e.g., OD2, iLabs, Sejer, etc.). XML data will enable lay-out flexibility on the third party distributor website.
<b>Asynchronous actions</b>	none
<b>Design suggestions</b>	none
<b>Issues</b>	None.

#### 15.4.3 Available resources listing

<b>UCId</b>	UC15.4.3
<b>Use case</b>	Available resources listing
<b>Description</b>	This Use Case describes a typical resources listing performed by a student through the distribution portal available in its school's VLE . After logging in, the portal only displays the content he already bought, for rapid access.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has an account on the content distribution system ; his client environment is properly configured to list his available content after logged in.

<b>Steps</b>	1. The User logs in the system through the AXMEDIS login client 2. Once login is accepted, the client displays the list of AXMEDIS object available to this user
<b>Post-conditions</b>	Once logged in, each time the user reaches the portal, the list of resources available to him is displayed
<b>Variations</b>	
<b>Asynchronous actions</b>	
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 15.4.4 Content Access

<b>UCId</b>	UC15.4.4
<b>Use case</b>	Catalogue Content Access
<b>Description</b>	The user selects the Catalogue content
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has selected content from a Catalogue listing
<b>Steps</b>	1 The user accesses the Catalogue Content page with editorial and product information 2 The user is prompted with two options: pre-download or purchase 3 If user opts for pre-download the user clicks on the AXMEDIS Object reference URL in order to fetch the content. If user opts for immediate purchase the user is sent to the MediaClub acquisition procedure for the AXMEDIS Object selected
<b>Post-conditions</b>	Depending if content is free of charge or requires a transaction, the user will be directed open the AXMEDIS Object or to the MediaClub Payment Gateway
<b>Variations</b>	This AXMEDIS Content Access could be managed by third party distributor (e.g., OD2, iLabs, Sejer, etc.). XML data will enable lay-out flexibility on the third party distributor website.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	Catalogue Content page may include editorial text, picture data
<b>Issues</b>	None.

#### 15.4.5 User Page

<b>UCId</b>	UC14.4.5
<b>Use case</b>	User Profiling
<b>Description</b>	
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has successfully registered in order to access the MediaClub
<b>Steps</b>	1 user accesses MediaClub user page 2 user selects product profiling option 3 user selects content preferences 4 user can view all content purchased, transactions, validity of licenses 5 user can view suggested content
<b>Post-conditions</b>	On subsequent access to the MediaClub user is prompted with customized pages that are assembled based on content preferences and on marketing profile
<b>Variations</b>	The user may re-run the procedure to update his content preferences.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	(TBD)
<b>Issues</b>	Legal disclaimer for privacy

## 15.5 Catalogue Content Purchase

### 15.5.1 Content Fetching

<b>UCId</b>	UC15.5.1
<b>Use case</b>	Content Fetching
<b>Description</b>	<p>As the user selects content fetching the AXMEDIS plug-in opens and Content delivery starts. User can select the 3 different delivery modes:</p> <ul style="list-style-type: none"> <li>• <b>Streaming.</b> Similar to a broadcast experience, user acquires license and subsequently starts streaming content. Recommended only for higher bandwidth (450kb/s or above).</li> <li>• <b>Download.</b> After acquiring a license, the user can download the media (up to 10Mb/s encoding). Media can be viewed from the user's computer after the downloading process (can take 1-8 hours according to user access)</li> <li>• <b>Pre-Download.</b> User can first download content and then is prompted to purchase license.</li> </ul> <p>The user can check any time that the progress bar, indicating the download state, is advancing.</p>
<b>Actors</b>	The Content Consumer (user) AXMEDIS plug-in
<b>Assumptions</b>	The user has selected an AXMEDIS Object distributed in the Content Catalogue. This may happen directly after catalogue content access or after Catalogue Content transaction.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user selects delivery mode: pre-download, download, progressive download, streaming</li> <li>2 The AXMEDIS plug-in opens and content delivery starts according to the delivery mode chosen by the user</li> <li>3 The user opens the jobs panel where all current downloads are displayed</li> <li>4 The user reads the remaining time for the end of transmission</li> <li>5 The user can open the folder where the content is being received</li> <li>6 The user can interrupt the reception of a given content</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	<p>The user, after opening the folder where the content is being received, deletes an incomplete and/or temporary file. This could put the AXMEDIS Client Application in an inconsistent state.</p> <p>The user may also activate a previously purchased license while fetching content in progressive download.</p>
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

### 15.5.2 User Authentication Form

<b>UCId</b>	UC15.5.2
<b>Use case</b>	User Authentication Form
<b>Description</b>	The user will be requested to authenticate in order to start any content fetch or transaction
<b>Actors</b>	The Content Customer (user) (involved in the purchase/rental operation) The MediaClub (entity performing all required checks to ensure that purchase/rental operations are valid and legal)
<b>Assumptions</b>	The user has access to the Catalogue

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user enters his identification information (this does not necessarily mean personal details, it will be sufficient to have proper credentials, e.g., login/password)</li> <li>2 The user credentials are sent to the MediaClub for verification</li> <li>3 The user waits for the server response</li> <li>4 If the user is identified as a regular one permission to proceed is granted and user can access all restricted areas of the Mediaclub that enable to fetch, purchase and acquire licenses for content, otherwise purchase procedure is aborted and user is sent back to browsing</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	This Authentication Form could be published by third party distributor (e.g., OD2, iLabs, Sejer, etc.). XML data will enable lay-out flexibility on the third party distributor website.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None.
<b>Issues</b>	None

### 15.5.3 Catalogue Content Transaction

<b>UCId</b>	UC15.5.3
<b>Use case</b>	Catalogue Content Transaction
<b>Description</b>	The user is prompted with multiple payment options. Te user confirms the intention of purchasing the selected AXMEDIS Content. The user provides payment related information along with data needed to ensure legal validity of requested operation.
<b>Actors</b>	The Content Consumer (user) The MediaClub Payment Gateway
<b>Assumptions</b>	The user has selected the Catalogue content
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The MediaClub Payment Gateway shows to the user all billing information available including: <ul style="list-style-type: none"> <li>o Price</li> <li>o Conditions for each selected item</li> <li>o Related use licence</li> <li>o Scope and limitations</li> <li>o Possible constraints</li> </ul> </li> <li>2 The MediaClub Payment Gateway asks the user to verify and accept presented terms</li> <li>3 If the user accepts procedure continues otherwise is aborted and user is sent back to browsing</li> <li>4 The user shall finalise billing information</li> <li>5 Once billing information are provided the user is requested to select the payment method (credit card, electronic wallet, pre paid card, pre assigned tokens or similar)</li> <li>6 The MediaClub Payment Gateway requires clearance to the AXMEDIS Distributor for the provided payment ID.</li> <li>7 If payment ID is cleared the user will be charged the cost</li> <li>8 The MediaClub Payment Gateway provides the system the proper clearance and the license delivery is authorized.</li> <li>9 The user receives confirmation of transaction OK on a web page</li> <li>10 The user receives an email notification that transaction has been succesful</li> <li>11 User can start fetching content and come back subsequently in the user page for license activation. Alternatively the user can immediately activate license and start viewing content during content fetching</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step

<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	A supplementary actor could be a bank or other institution that will handle the money transaction and has to be a third trusted party for both the user and the AXMEDIS Certifier.
<b>Issues</b>	None.

#### 15.5.4 Content Access

<b>UCId</b>	UC15.5.4
<b>Use case</b>	Content Access
<b>Description</b>	The user accesses his local cache containing several AXMEDIS Objects.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The AXMEDIS Content is successfully received.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user accesses the AXMEDIS Object for playing it</li> <li>2 The AXMEDIS Object is delivered to either the AXMEDIS Viewer or the standard application (with an additional AXMEDIS plug-in)</li> <li>3 The application detects if the Object needs to acquire a license</li> <li>4 The application finds a pre-acquired license for the Object and play it</li> <li>5 The application needs a new license for the Object and tries to contact the MediaClub.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 15.5.5 Content Preview

<b>UCId</b>	UC15.5.5
<b>Use case</b>	Content Preview
<b>Description</b>	The user browses one/more AXMEDIS Object(s). The user opens and plays some short previews (if they are available) integrated with the received AXMEDIS Object. The user decides to buy or not the received AXMEDIS Content.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The AXMEDIS Object has been integrally received.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the AXMEDIS Object locally stored in his local cache</li> <li>2 The user browses the AXMEDIS Object, using the AXMEDIS Info associated to the Object</li> <li>3 The user reaches a preview available for the Object</li> <li>4 The user plays the AXMEDIS Object Preview</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	One or more previews (depending on the internal structure of the AXMEDIS Object) should be available for the final user, in order to help him in the content evaluation before purchasing it.
<b>Issues</b>	None.

#### 15.5.6 License Acquisition

<b>UCId</b>	UC15.5.6
-------------	----------

<b>Use case</b>	License Acquisition
<b>Description</b>	The user receives a license for playing the content
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user is logged-in to the MediaClub The user has selected to play an AXMEDIS content
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the protected part of the AXMEDIS Object</li> <li>2 The Object is delivered to the application/viewer charged to open/play it</li> <li>3 The Application/Viewer has an internal plug-in able to detect if the Object to open needs a license</li> <li>4 The AXMEDIS Viewer, using the internal plug-in, contacts the MediaClub in a protected mode (a secure connection is established with the MediaClub)</li> <li>5 The MediaClub authorizes the AXMEDIS Certifier and Supervisor to provide the user with a license corresponding to the business rule associated to product purchased by the user</li> <li>6 The user receives the AXMEDIS license useful to open the protected part of the AXMEDIS Object</li> <li>7 The user receives a confirmation page that license has been successfully issued</li> <li>8 The user consumes the AXMEDIS Object following the rules contained in the AXMEDIS license</li> </ol>
<b>Post-conditions</b>	The user plays the content
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None.
<b>Issues</b>	Security, privacy and transparency are key requirements.

#### 15.5.7 Multi-device license activation and back-up

<b>UCId</b>	UC15.5.7
<b>Use case</b>	Multi-device license activation and back-up
<b>Description</b>	The user copies some interesting content in a device other than initial PC
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The device must be supported by the AXMEDIS Client plug-in Any Content copy or backup has to be expressly authorized in the license terms.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the copy/backup interface of the AXMEDIS Client plug-in</li> <li>2 The user selects all Objects involved in the copy operation</li> <li>3 The user specifies the device where the AXMEDIS Content has to be copied.</li> <li>4 the user can start a new license activation procedure (if he has right to activate license on new device) or else purchase new license for new device</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None.

#### 15.5.8 Pre-ordering and registration for a group of students

<b>UCId</b>	UC15.5.8a
<b>Use case</b>	Pre-ordering for a group of students
<b>Description</b>	This case describe the specific procedure used by a “master user” (teacher) to buy content licenses for multiple other user.
<b>Actors</b>	The teacher
<b>Assumptions</b>	The teacher has to be registered in the system. The students (users) may not be registered in the system ;

<b>Steps</b>	<ol style="list-style-type: none"> <li>1.The teacher orders the product and request N licenses</li> <li>2.Through the commercial service, or a web server, or a pre-registered account etc, the teacher pays the bill</li> <li>3. Pre-ordering is saved in the Pre order database, waiting for activation</li> <li>4. The teacher receives an e-mail confirming its order, and containing the Activation Number</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	Parameters given at pre-ordering time may change the kind of license waiting for activation : license per user/product, license per device, license per domain
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None.

<b>UCId</b>	UC15.5.8b
<b>Use case</b>	Automatic registering for a group of students
<b>Description</b>	This case describe the automatic registering of students (user) when they use the content pre-ordered by their teacher for the first time
<b>Actors</b>	The teachers The students
<b>Assumptions</b>	The teacher has pre-ordered enough licenses for all of its students ; The students are not yet registered in the system The students are using a dedicated client
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The teacher gives to the students the URL to access the content ;</li> <li>2 The teacher gives the Activation Number for this content to the students ;</li> <li>3 Through the dedicated client, the student access the URL given by the teacher</li> <li>4 The system ask the student for the Activation Number for this product</li> <li>5 The student enter the Activation Number</li> <li>6 The dedicated client associate automatically computed user/device identification data and send them along with the Activation Number</li> <li>7 The system creates an AXMEDIS user/device with the identification data</li> <li>8 The system creates an AXMEDIS License corresponding to the parameters given at pre-ordering time, for the previously created user/device</li> <li>9 The license is made available trough AXMEDIS for the user to be able to view the requested content</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	Depending on the initial parameters, the license is granted for a user, for a device or for a combination of both.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None.

## 15.6 Business Models

### 15.6.1 Rental

<b>UCId</b>	UC15.6.1
<b>Use case</b>	Rental
<b>Description</b>	User pays to view a media in streaming or download mode before he can access to it. After having acquired a license for a media download, this license remains valid for a limited time before it expires. After having rented a media in streaming or download mode, the user will be able to see it as often as he likes within the validity period.
<b>Actors</b>	The Content Consumer (user)

<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 the customer chooses the content</li> <li>2 he provides minimal personal information, chooses the payment</li> <li>3 the distributor confirms the successful payment transaction</li> <li>4 the customer downloads the content</li> <li>5 after the content expiration date the content is not accessible anymore</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	(TBD)
<b>Issues</b>	None.

### 15.6.2 pay per download

<b>UCId</b>	UC15.6.2
<b>Use case</b>	pay per download
<b>Description</b>	The end user pays to download a content that will be seen only once.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 the customer chooses the content</li> <li>2 he provides minimal personal information, chooses the payment</li> <li>3 the distributor confirms the successful payment transaction</li> <li>4 the customer downloads the content</li> <li>5 if the customer wants to access again to the content he has to replicate this procedure</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	(TBD)
<b>Issues</b>	None.

### 15.6.3 Sell-through

<b>UCId</b>	UC15.6.3
<b>Use case</b>	Sell-through
<b>Description</b>	User acquires a permanent license and owns the media after the download, just as a purchased DVD (including the right to watch it without limitations and to burn it). Equivalent to a content offer of the shelf (CD, DVD, book etc.)
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	like pay per download
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	(TBD)
<b>Issues</b>	None.

### 15.6.4 subscription

<b>UCId</b>	UC15.6.4
<b>Use case</b>	subscription

<b>Description</b>	Based on a recurrent fee. User purchases either full access to a set of contents that can be viewed throughout the period he pays for. Or he gets a defined number of credits every month for that the subscription is valid.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 the customer chooses the subscription</li> <li>2 provide personal information, chooses the payment system and the invoice method (paper mail)</li> <li>3 the distributor confirms the subscription success and the customer can start using the service</li> <li>4 the renewal is automatically done until the customer terminates the subscription</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	(TBD)
<b>Issues</b>	None.

#### 15.6.5 pay per minute

<b>UCId</b>	UC15.6.5
<b>Use case</b>	pay per minute
<b>Description</b>	The customer is charged for the time the content is streamed
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 the customer chooses the content</li> <li>2 he provides personal information, chooses the payment and invoice method (mail / paper)</li> <li>3 the distributor confirms the successful payment transaction</li> <li>4 the customer can start see/use the content</li> <li>5 based on the recurrency defined by the distributor, the customer receives the invoice and he is automatically charged for the number of minutes where he used the content</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXMEDIS has to provide information of the minutes required
<b>Issues</b>	None.

#### 15.6.6 pay per Kb downloaded

<b>UCId</b>	UC15.6.6
<b>Use case</b>	Pay per Kb downloaded
<b>Description</b>	The customer is charged for the Kb downloaded
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	same procedure as pay per minute where the measure unit is the Kb instead of the minutes
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXMEDIS has to provide information of the minutes required

<b>Issues</b>	None.
---------------	-------

### 15.6.7 pay per day

<b>UCId</b>	UC15.6.7
<b>Use case</b>	pay per day
<b>Description</b>	The customer is charged for the number of days he access to the content
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	same procedure as pay per minute where the measure unit is the number of days when the customer use the content instead of the minutes
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	AXMEDIS has to provide information of the minutes required
<b>Issues</b>	None.

### 15.6.8 pay per credits

<b>UCId</b>	UC15.6.8
<b>Use case</b>	pay per credits
<b>Description</b>	All contents are associated to a set of credits which are translated into currency in the case of PPV, or are visualized as mere credits for prepaid users. example implementation could ser an equivalence of <b>100 credits = 1 euro</b> . Technically speaking the users always purchases a set of credits. This enables to provide ease of communication for all offers with users able to easily asses the value of their purchase. Credits provide also and easy mean of negotiation with content owners. Experience with music prepaid credits show that the model is very adapted to Internet use.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	See catalogue Content Transaction
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 User can purchase in advance (prepaid) credits.</li> <li>2 Every time he rents or buys a media, a certain number of credits will be deducted from his account.</li> <li>3 Prepaying a higher number of credits results in a volume discount for the user</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	
<b>Issues</b>	None.

### 15.6.9 Grouped licenses

<b>UCId</b>	UC15.6.9
<b>Use case</b>	Grouped Licenses

<b>Description</b>	As part of some packaged offers (e.g. a VLE(1) provider wants to include educational content in his offer for a school/group of school), license for a group of products can be granted for whole schools for multiple school years. (1) VLE = Virtual Learning Environment
<b>Actors</b>	VLE Provider, Content Provider, Users
<b>Assumptions</b>	VLE Provider includes AXMEDIS client into his VLE offer
<b>Steps</b>	1. VLE Provider concludes a deal with Content Provider for including N products with specific license conditions into his VLE offer 2. Content Provider create the licences corresponding to the grouped offer : mostly license for domains, where a domain can be a school, group of school 3. Inside the domain, the Users can access the contents
<b>Post-conditions</b>	
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	
<b>Issues</b>	Precise the concept of “domain”

### 15.6.10 Packaged offers

<b>UCId</b>	UC15.6.10
<b>Use case</b>	Packaged offers
<b>Description</b>	On one distribution portal, multiple content providers associate themselves to provide one priced packaged offer. The package contains objects from each content provider, each under specific licensing conditions
<b>Actors</b>	Distributor, Contents providers
<b>Assumptions</b>	
<b>Steps</b>	1. Multiple Content providers agrees on a packaged offer including some of their products under specific licensing terms at an agreed price, for a specific distribution channel 2. The distributor creates the package on its portal 3. Customer buys the package on the portal 4. The distributor split the money between content publishers present in the packaged offer, according to predefined rules 5. Content provider create licenses according to the package content
<b>Post-conditions</b>	
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	
<b>Issues</b>	None.

## 15.7 Advanced Payment methods

### 15.7.1 Gift Certificates

<b>UCId</b>	UC15.7.1
<b>Use case</b>	Gift Certificates
<b>Description</b>	Gift certificates allow a customer to buy a credit and to gift them to another customer. The credit is bought in a shop and can be used only in that shop.
<b>Actors</b>	The Content Consumer (end user);
<b>Assumptions</b>	

<p><b>Steps</b></p>	<p><b>Definition</b>  <i>CustomerA</i>: buys a credit for a friend  <i>CustomerB</i>: is the friend who receive the gift</p> <p><b>Procedure to Purchase a Gift Certificate</b>  <b>Step 1: start</b>  In the web site shop <i>customerA</i> clicks on a link 'Buy a gift certificate'. This link is part of the portal.</p> <p><b>Step 2: payment details</b>  The distributor application asks to the customer:</p> <ul style="list-style-type: none"> <li>• the amount to buy</li> <li>• mail address of the friend</li> <li>• payment <span style="float: right;">details</span>  (these information are stored by the Distributor adding a PIN code)</li> </ul> <p><b>Step 3: mail delivery to <i>customerB</i></b>  An application send an email to <i>customerB</i> providing information about the gift and how to redeem it. This text can be partially typed by <i>customerA</i>  Instruction contains a link to the website where <i>customerB</i> can redeem the gift and (embedded in the link) a PIN code that will be burn once the customer redeem the gift.</p> <p><b>Procedure to redeem the gift certificates</b>  <i>CustomerB</i> clicks on the link present in the mail reaching the Distributor application that recognize the PIN and knows the credit related; the credit is shown to the customer inviting him to start the standard purchase procedure (selection of staff to buy and ok to the kart content)</p> <p>When <i>customerB</i> approves the Kart content, there is a control about the amount to pay and the value of the kart with 3 different situation:</p> <p>1) gift value=value to purchase  the customer sees a confirmation page + receives an email</p> <p>2) gift value&gt;value of the kart  the customer can use the credit available in following purchases. Technically the value of the PIN code assigned to the customer is decreased</p> <p>ex. Gift certificate value = 50 € (that is the value associated to the PIN generated for that gift)  <i>customerB</i> buys 30 € in Tiscali music club  the new value of the PIN code is 20 € available for new purchases</p> <p>the customer sees a confirmation page reminding the credit available + receives an email with the link where to redeem the credit available</p> <p>3) gift value&lt;value of the kart</p> <p>the customer is required to chose a payment method to pay the difference or to come back to the kart to remove some items.</p>
<p><b>Post-conditions</b></p>	<p>The gift certificates has an expiration date  The distributor application supports the possibility for the customer to check the gift certificates already available</p>

<b>Variations</b>	None.
<b>Asynchronous actions</b>	?
<b>Design suggestions</b>	The gift certificates application is stand alone
<b>Issues</b>	None.

### 15.7.2 Wallet

<b>UCId</b>	UC15.7.2
<b>Use case</b>	Wallet
<b>Description</b>	<p>Wallet is a payment account the customer opens with the Distributor for paying transactional services.</p> <p>The wallet creation needs a first deposit by the customer and can be used immediately. The wallet can be recharged with following deposits.</p> <p>The wallet saves the payment method used by the customer that is proposed for following deposit.</p>
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	
<b>Steps</b>	<p><b>Wallet creation</b></p> <p>The customer in the distributor application ask for wallet registration providing authentication information (mail and password) and receives a security key to be used for all the transactions.</p> <p>Afterwards made the first deposit using the payment methods allowed by the distributor</p> <p><b>Wallet ecare</b></p> <p>The customer can:</p> <ul style="list-style-type: none"> <li>• Check the balance</li> <li>• Recharge</li> <li>• Check the statement (List of deposits, List of the purchases done)</li> <li>• Change the secure key</li> <li>• Change payment method used</li> </ul> <p>To access to the wallet ecare, to make payments, and to recharge the wallet, the security key is always requested.</p> <p><b>Payments</b></p> <p>if the customer decides to pay with wallet and the balance is NOT enough to cover the new purchase he is asked for recharge.</p> <p><b>Wallet termination</b></p> <ul style="list-style-type: none"> <li>• Expiration</li> </ul> <p>The credit of the customer expires after a period defined by the distributor. After this period the customer credit is flagged as 'suspended' but can be used by the customer making another deposit.</p> <ul style="list-style-type: none"> <li>• Termination</li> </ul> <p>The customer can ask to close his wallet removing the payment information. No refund applicable</p>
<b>Post-conditions</b>	
<b>Variations</b>	None.

<b>Asynchronous actions</b>	
<b>Design suggestions</b>	
<b>Issues</b>	None.

## 16 AXMEDIS for Distribution towards Mobiles

### 16.1 General Assumptions and Notes to Architecture

- 1) The AXMEDIS enabled ILABS, IRC distribution system includes:
  - a) An AXMEDIS network node, which:
    - i) Automatically fetches all AXMEDIS objects matching pre-set criteria; licensing attributes, content type, time-span, etc.
    - ii) Makes all fetched content and assets available for immediate use, providing online availability of ready-to-use files in specific formats (WMA, MIDI, etc).
    - iii) Maintains a list of all files available for use from local storage.
    - iv) Automatically synchronizes object and content expiration, and license changes with the AXMEDIS network.
  - b) The ILABS, IRC APS (Application Server), with integrated Personalization (PE) and Handset Management engines (HME).
  - c) A plug-in that interacts with the AXMEDIS platform, encapsulating and simplifying the platform functionality for the ILABS, IRC servers and components.
- 2) The AXMEDIS enabled ILABS, IRC Transcoding Server includes:
  - a) A Transcoding Server, which manages the transcoding logic and routines.
  - b) A plug-in that interacts with the AXMEDIS platform, encapsulating and simplifying the platform functionality for the ILABS, IRC servers and components.
  - c) A Transcoding platform including Codecs, configuration and Interface.
- 3) Categories:
  - a) Category: this object is defined within Mobile Application. It is meant as an "area of interest" and it consists of:
    - i) a category key, that has to be unique inside the Mobile Application
    - ii) a sequence of category names, one for each language supported by the Mobile Application; these names are the ones that will be seen by Mobile Application users
    - iii) an associated query to retrieve contents for category

### 16.2 Use Cases

#### 16.2.1 Domain registration

<b>UCId</b>	UC16.2.1
<b>Use case</b>	Domain registration
<b>Description</b>	The Mobile Admin creates and registers a new domain within the AXMEDIS system
<b>Actors</b>	Mobile Admin, Mobile Front End, Mobile Back End, Domain Management module, PMS Domain, PMS Communication module
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin accesses to a reserved section of the Portal and generates a new domain identifier</li> <li>2 The Mobile Front End forwards the identifier to the Mobile Back End</li> <li>3 The Mobile Back End forwards received data to the Domain Management module</li> <li>4 The Domain Management module prepares a "Domain object" that will be passed to the PMS Domain through the PMS Communication module in order to register the newly created domain within the AXMEDIS system</li> <li>5 The PMS Domain registers the newly created domain within the AXMEDIS system</li> </ol>

<b>Post-conditions</b>	The new domain is registered and operational
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.2 Content Preparation/ingestion

<b>UCId</b>	UC16.2.2
<b>Use case</b>	Content Preparation/ingestion
<b>Description</b>	AXMEDIS object are ingested and packed for delivery
<b>Actors</b>	Mobile Admin, ingestion procedure
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The mobile admin activates the ingestion procedure</li> <li>2 AXMEDIS object are packed into IMS Content Packages, where each item refers to the resource that is represented by AXMEDIS object.</li> </ol>
<b>Post-conditions</b>	Content is ingested and post-processed
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.3 Content Retrieving Criteria Management

<b>UCId</b>	UC16.2.3
<b>Use case</b>	Content Retrieving Criteria Management
<b>Description</b>	The Mobile Admin defines catalogue categories, which represent possible areas of interest, such as art, science, history and so on.
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin defines catalogue categories</li> <li>2 For each category: <ol style="list-style-type: none"> <li>2.1 Assigns a “category name”</li> <li>2.2 Defines related retrieving query.</li> <li>2.3 Defines active categories (that means visible by users).</li> </ol> </li> </ol>
<b>Post-conditions</b>	Updated content retrieving criteria are operational
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	By default, a new created category is not active; it has to be explicitly activated

### 16.2.4 Content Retrieving Criteria Definition

<b>UCId</b>	UC16.2.4
<b>Use case</b>	Content Retrieving Criteria Definition
<b>Description</b>	The Mobile Admin defines content retrieving definitions and related queries
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin interacts with the Front-End which to start the content retrieving criteria definition activity</li> <li>2 The request is forwarded to the Back End that replies with a Criteria definition query form</li> <li>3 The mobile manager fills requested data and confirms</li> <li>4 Inserted data are checked by the Mobile Back-End and forwarded to the Catalogue Management module</li> <li>5 The Catalogue management module interacts with the Database Management module in order to store queries.</li> </ol>
<b>Post-conditions</b>	Updated content criteria definition are available
<b>Variations</b>	Catalogue Management module could cache categories queries in order to avoid communication operations with database to retrieve them later
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	<p>A category requires a unique key within the whole categories set since it is used to retrieve corresponding query</p> <p>The Mobile Admin is also requested to give translation of category name for each language supported by the system since users won't see category key but a category name properly translated</p>

### 16.2.5 Content Retrieving Criteria Selection

<b>UCId</b>	UC16.2.5
<b>Use case</b>	Content Retrieving Criteria Selection
<b>Description</b>	The Mobile Administrator decides which of the categories previously defined and stored into the system are actually seen by users
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Administrator asks to the Front End the list of previously defined categories</li> <li>2 The request is dispatched to the Back End and Catalogue Management modules</li> <li>3 The latter retrieves from database the list of categories and their current status (active/ not active)</li> <li>4 The generated List is passed back and shown to the Mobile Admin</li> <li>5 The mobile admin chooses whether updating or not categories status and then sends confirmation (whenever needed).</li> <li>6 The Categories status (active/not active) is then updated in the database.</li> </ol>
<b>Post-conditions</b>	Selected content retrieval criteria are operational
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.6 Content Retrieving Criteria Removing

<b>UCId</b>	UC16.2.6
<b>Use case</b>	Content Retrieving Criteria Removing
<b>Description</b>	The Mobile Administrator removes content selection criteria previously stored into the system.
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin through the Front End requests to access to the list of inserted criteria</li> <li>2 The request is handed over to the Back End module that recovers data from the database</li> <li>3 The Mobile Admin selects the criteria to delete and confirms</li> <li>4 The request is processed and selected criteria are removed.</li> </ol>
<b>Post-conditions</b>	Selected content retrieval criteria are no more operational
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.7 Supported device profile adding

<b>UCId</b>	UC16.2.7
<b>Use case</b>	Supported device profile adding
<b>Description</b>	The Mobile Admin adds a profile for a newly supported device
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin requests insertion of a newly supported device profile to the Mobile Front End</li> <li>2 The Mobile Front End forwards the request to the Mobile Back End which in turns returns the data structure to be filled in.</li> <li>3 The Mobile Admin fills all required form fields and submits insertion to the Front End.</li> <li>4 The Request is again forwarded to the Back End, which performs basic data checking including one with the User Management.</li> <li>5 The User Management module communicates the request to the Device Profile Management</li> <li>6 The Device Profile Management creates a Device Profile object from inserted data and asks for inserting it into local database.</li> <li>7 The storage is performed and a batch procedure is launched to adapt available objects for new device profile.</li> </ol>
<b>Post-conditions</b>	New supported profile device are operational
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.8 Supported device profile removing

<b>UCId</b>	UC16.2.8
<b>Use case</b>	Supported device profile removing
<b>Description</b>	The Mobile Admin decides/needs to remove some of the supported device profiles previously defined.
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin asks the front end the list of supported device profiles</li> <li>2 The front end retrieves, thanks to backend, the list of supported device profiles</li> <li>3 The Mobile Admin chooses among them the one to delete and requires deletion.</li> <li>4 The request is forwarded to the backend that processes the request and removes that device profile (already adapted content is not deleted but simply marked as not useful)</li> </ol>
<b>Post-conditions</b>	Selected profile devices are no more operational
<b>Variations</b>	It is foreseen as possible that the Mobile Admin requires a combination of device profile and related adapted content deletion. In this case the procedure would be more complex as to be positively completed would also require content deletion completion
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Some kind of marking is required for contents associated to the removed device profiles. This is specifically needed to avoid loss of content and yet prevent useless computational effort during search and retrieval phases related to content offering preparation.

### 16.2.9 User registration by administrator

<b>UCId</b>	UC16.2.9
<b>Use case</b>	User registration by administrator
<b>Description</b>	The Mobile Admin registers a new user
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin requests the front end to retrieve the registration form</li> <li>2 Once retrieved the Mobile Admin fills-in all mandatory data and confirms</li> <li>3 The front end module forwards imputed data to the user management module</li> <li>4 The user management module forwards needed data to the AXCS.</li> <li>5 The AXCS registers the new user and provides a positive ACK</li> <li>6 The positive confirmation is sent back to the Mobile Admin</li> <li>7 The Mobile Admin sends a formal registration notice to the requesting new user</li> </ol>
<b>Post-conditions</b>	Specified user is now registered
<b>Variations</b>	The user self-registers.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The present case is foreseen not just for the sake of completeness, but also because in some context (business context mainly) it is usually the practice to have a service subscription validation prior to the subscription. User registration by Mobile Admin could be activated, for example, via e-mail (request coming from the user willing to be registered).

### 16.2.10 User update by administrator

<b>UCId</b>	UC16.2.10
<b>Use case</b>	User update by administrator
<b>Description</b>	The Mobile Administrator updates users' data.
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin requests the user management module the list of registered users</li> <li>2 The Mobile Admin selects the user whose data needs update</li> <li>3 The Mobile Admin modifies and confirms user's data</li> <li>4 The new data are updated into the system</li> </ol>
<b>Post-conditions</b>	Specified user data have been updated
<b>Variations</b>	The user self-changes provided data.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	The present case is foreseen not just for the sake of completeness, but also because in some context (business contest mainly) it is usually the practice to have a service subscription validation prior to the subscription. User data update by Mobile Admin could be activated, for example, via e-mail (request coming from the user being registered).

### 16.2.11 User remove by administrator

<b>UCId</b>	UC16.2.11
<b>Use case</b>	User remove by administrator
<b>Description</b>	The Mobile Admin proceeds to remove a user from the system
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin requests the User management module to remove a user from the system,</li> <li>2 The user management module requests the AXCS to perform an update of the "finalRegDeadline" parameter into user's registration data</li> <li>3 The AXCS confirms the performed update process</li> <li>4 User's data are deleted from local database</li> <li>5 The Mobile Admin sends an email to the user to notify deletion.</li> </ol>
<b>Post-conditions</b>	Selected user is no more registered
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.12 User roles management

<b>UCId</b>	UC16.2.12
<b>Use case</b>	User roles management
<b>Description</b>	The Mobile Administrator can change a user's role, for example assigning also an administrator role (users can have more than one role at the time).
<b>Actors</b>	Mobile Admin
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Mobile Admin requests the user management module the list of registered users</li> <li>2 The Mobile Admin selects the user whose role data needs update</li> <li>3 The Mobile Admin modifies and confirms user's data</li> <li>1 The new data are updated into the system</li> </ol>
<b>Post-conditions</b>	Selected users' role has been positively updated/enhanced
<b>Variations</b>	Instead of adding a role to a user it could be removed
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None

<b>Issues</b>	None
---------------	------

### 16.2.13 User registration

<b>UCId</b>	UC16.2.13
<b>Use case</b>	User registration
<b>Description</b>	The User has to register before being able to use AXMEDIS Mobile Application.
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user interacts with the Application Front End in order to retrieve a registration form. There are two types of data to insert: the elements that should be filled by every user wishing to register (mandatory elements) and the elements that would be very useful whenever the user decides to fill in (recommended elements).</li> <li>2 The user inputs the data and confirms; the registration request is forwarded to the backend</li> <li>3 The backend performs some consistency and validity checks. If provided data are formally corrected they are passed to User Management module</li> <li>4 The User management module makes a first check to verify if the user is already present into the local DB to avoid useless duplications.</li> <li>5 The User management module sends a registration request to the AXCS via the AXCS Communication module, which function is to properly format and forward requests to the AXCS</li> <li>6 If the user registration is successful then user data are stored locally into local database.</li> <li>7 The user management module, at the end of the registration process, sends an email to the user as feedback to communicate the attributed AXMEDIS user identifier.</li> </ol>
<b>Post-conditions</b>	The user is successfully registered both locally and on AXMEDIS
<b>Variations</b>	The user has already registered himself into AXMEDIS through another distributor so s/he has to register only for the Mobile Application now. Therefore the Registration form should allow inserting also an AXUID if the user already owns one. In this case communication with AXCS is limited to the notification of the new subscribed distributor if not needed at all.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.14 Certification of users

<b>UCId</b>	UC16.2.14
<b>Use case</b>	Certification of users
<b>Description</b>	A user wishes to be certified (with own device) in order to access to AXMEDIS objects
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user requires to be certified (with own device) in order to access to AXMEDIS objects</li> <li>2 The front end fetches the request form from the backend and presents it to the user</li> <li>3 The user has to provide (directly or indirectly) the needed information to the application that forwards user and tool information to the Domain PMS</li> <li>4 The Domain PMS performs all necessary checks and operations</li> <li>5 In case of positive result, the Domain PMS returns acknowledge to the requesting application</li> <li>6 The front end receives the positive feedback and provides a proper feedback to the user.</li> </ol>
<b>Post-conditions</b>	The user (and related device) is properly certified
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Tool-user-device become linked once certified, as an AXMEDIS tool is certified for an AXMEDIS user on an AXMEDIS device.

### 16.2.15 Client application download

<b>UCId</b>	UC16.2.15
<b>Use case</b>	Client application download
<b>Description</b>	User downloads AXMEDIS Player on own devices in order to be able to use AXMEDIS contents
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Users logs onto the Mobile Application Portal</li> <li>2 The user requires download instructions to the front end</li> <li>3 The front end requires instructions provision to the backend</li> <li>4 Instructions are returned to the front end that takes care to present them to the user</li> <li>5 The user prepares the device for connection and download</li> <li>6 The user connects to “the network” (here this has to be regarded in the wider sense of the term) using his personal device and download the AXMEDIS client application</li> </ol>
<b>Post-conditions</b>	The AXMEDIS player is downloaded on the user device
<b>Variations</b>	A more complete case foresee the download, install and usage
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.16 User login

<b>UCId</b>	UC16.2.16
<b>Use case</b>	User login
<b>Description</b>	A user logs to the Mobile Application Portal
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user wishes to use AXMEDIS Mobile functionalities</li> <li>2 The user performs a login to the mobile portal by accessing the login form and providing login and password chosen at registration time</li> <li>3 The Mobile front end receives the imputed data and passes it to the Mobile Back End</li> <li>4 The Mobile backend performs some basic checks and then further forwards received data to the User Management module</li> <li>5 The user management module requests the User Database Management module to verify user credentials</li> <li>6 Upon confirmation of successful data match, access to the Mobile Portal is granted</li> </ol>
<b>Post-conditions</b>	The user is properly logged onto the mobile portal
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.17 User interface language selection

<b>UCId</b>	UC16.2.17
<b>Use case</b>	User interface language selection
<b>Description</b>	A user wants to change the current language used by the application
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user interacts with the Front End requesting a change in GUI language</li> <li>2 The front end forwards the requested change to the Back End</li> <li>3 The list of supported languages is retrieved and passed back to the front end</li> <li>4 The front end displays the list of supported languages</li> <li>5 The user chooses a language and from this point onwards all pages will be visualized using the selected language.</li> </ol>
<b>Post-conditions</b>	The GUI is in the selected language
<b>Variations</b>	If a user is already logged then the currently selected language is updated also into user's profile, elsewhere language changes only for visualization.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	At any time a user can change the current language used by the application, yet this may imply some change in the usage scenario which may imply (but not necessarily) a potential thread as a change in interface during operation is far less usual (as a behaviour) than at initial (login) time and this could be due to a change in person performing the operation.

### 16.2.18 Catalogue loading and browsing

<b>UCId</b>	UC16.2.18
<b>Use case</b>	Catalogue loading and browsing
<b>Description</b>	A user wants to access contents browsing the Mobile Catalogue and related categories
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user wishes to use AXMEDIS Mobile functionalities to access to content</li> <li>2 The user performs a login to the mobile portal</li> <li>3 Once granted access the front end requires the back end to provide the list of available content categories</li> <li>4 Requests for content categories received at the back end are dispatched to the Catalogue Management module and categories are retrieved from the local database.</li> <li>5 Retrieved list of foreseen content categories is sent back to the front end that takes care to display it to the user</li> <li>6 The user chooses a category, and such information (received by the front end) is passed to the back end</li> <li>7 Related contents are retrieved both in the Lobster and via Query Support (and related communication modules).</li> <li>8 A list of the available content is generated and sent back to the front end that presents it to the user</li> <li>9 The user is now able to browse the list of available content (in the selected category)</li> </ol>
<b>Post-conditions</b>	The user is able to select content for purchase
<b>Variations</b>	<ol style="list-style-type: none"> <li>1 The Catalogue Management module could cache locally queries associated to categories in order to avoid retrieving them from database every time they are needed</li> <li>2 In order to achieve better performances, the Catalogue Management could cache contents data in order to be able to answer directly to further requests and to avoid calling the Database Management and the Content Retriever modules functionalities again later. This certainly applies to non AXMEDIS data, on the other hand for AXMEDIS data references used for retrieval in local AXDB will be cached in order to speed up the retrieval process as described previously.</li> </ol>
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	

### 16.2.19 Contents Search

<b>UCId</b>	UC16.2.19
<b>Use case</b>	Contents Search
<b>Description</b>	A user wants to get more contents than the ones available and listed on the Mobile Portal by categories selection, therefore performs a search
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user may wish to get more contents than the ones available and listed on the Mobile Portal by categories selection</li> <li>2 To retrieve other contents the user can exploit the Search Contents page and fill-in query form parameters.</li> <li>3 The query is dispatched locally and remotely through the Query on Demand module.</li> <li>4 Received results are turned into an object list by the backend</li> <li>5 The retrieved content list is presented to the user by the front end</li> </ol>
<b>Post-conditions</b>	The user is able to select content for purchase
<b>Variations</b>	
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	

**16.2.20 Getting content information**

<b>UCId</b>	UC16.2.20
<b>Use case</b>	Getting content information
<b>Description</b>	The user asks for more information about a specific content
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A user is interested in a specific content found on any Portal pages</li> <li>2 The user asks for more information about that content</li> <li>3 The request is received by the front end module that dispatches to the Catalogue Management via the backend</li> <li>4 The catalogue management retrieves content metadata from the local database or remotely through the Query Support</li> <li>5 Retrieved data are sent back to the front end for display to user</li> <li>6 The front end presents the user received data</li> </ol>
<b>Post-conditions</b>	The user is able to access to content info
<b>Variations</b>	Catalogue Management module already cached contents on catalogue loading, therefore no forwarding of requests is needed towards the Content Retriever and Query Support modules.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	

**16.2.21 Content Preview**

<b>UCId</b>	UC16.2.21
<b>Use case</b>	Content Preview
<b>Description</b>	User wants to have a preview of a specific content before purchase
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning, a content preview is available (either as a specific option or as a capability of the player)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The User is logged in and currently browsing the catalogue</li> <li>2 The user wants to have an idea on a specific content before purchasing a fruition license.</li> <li>3 The user asks for a “preview” of the content</li> <li>4 The front end receives the request</li> <li>5 The front end asks the AXMEDIS Player on the user device to display content preview</li> </ol>
<b>Post-conditions</b>	The user has been able to preview content whenever possible
<b>Variations</b>	The player is not able to perform a preview and therefore specific “preview” objects have been prepared at ingestion time
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	It is assumed that a preview will basically be a “piece” of the content to view/play/other depending on the type of the content and the possibility granted by the copyright regulations managed via the DRM, this could either be achieved directly at client player side by limiting access to resources or by producing specific “preview” objects that will be available on the catalogue for preview purpose only (those objects may be generated at ingestion time using the AXCP and a set of predefined specific formatting templates designed purposely for preview activities).
<b>Issues</b>	None

**16.2.22 Content Delivery**

<b>UCId</b>	UC16.2.22
<b>Use case</b>	Content Delivery
<b>Description</b>	Content is delivered to the user according to acquired license rights
<b>Actors</b>	User, user device (in case), mobile application
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The Front End interacts with the Back End to activate the delivery</li> <li>2 The back end forwards requests to the Content Management and Retriever modules.</li> <li>3 Content is retrieved either from the AXDB</li> <li>4 Also User Profile and Device Profile are retrieved as they are needed to have a format suitable for user device and preferences</li> <li>5 Content is adapted via AXCP</li> <li>6 Adapted content is delivered</li> </ol>
<b>Post-conditions</b>	The user has been able to have the selected content delivery
<b>Variations</b>	Content is retrieved from the Lobster as it not an AXMEDIS object
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 16.2.23 Content Acquire

<b>UCId</b>	UC16.2.23
<b>Use case</b>	Content Acquire
<b>Description</b>	A user wants to acquire rights to execute some actions over an object (install, uninstall, play, modify, etc...).
<b>Actors</b>	User
<b>Assumptions</b>	The system is properly functioning and the payment related modules are properly interconnected and operations. Is also assumed that all operation related to payments are positively terminated throughout the process.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user requests the front end to purchase a content</li> <li>2 The front end requests the back end the PARs list for the selected content</li> <li>3 The back ends retrieves the requested data from the Catalogue Management module</li> <li>4 Retrieved data are presented back to the user</li> <li>5 The User chooses the set of rights to acquire from the provided list</li> <li>6 The user is requested to confirm performed choice</li> <li>7 Upon conformation the user is requested to provide needed payment data. As some of these data have already been provided at registration time, the user is asked either to confirm the purchase form</li> <li>8 The user confirms</li> <li>9 The order confirmation is forwarded to the e-commerce module</li> <li>10 The E-Commerce module interfaces with the Bank payment module.</li> <li>11 If payment is successful then the Catalogue Management module asks to the PMS Communication module to forward to the PMS Web Service a license generation request.</li> <li>12 The PMS Communication module has to call three PMS methods in order to create a license: an initialization method, a method to add grants to license (one method call for each grant in license) and a finalization method, which returns new license identifier.</li> </ol>
<b>Post-conditions</b>	The user has been able to purchase the desired content (and related fruition)
<b>Variations</b>	As some of data related to content acquisition payments have not been provided at registration time, the user is asked to complete the purchase form
<b>Asynchronous actions</b>	None

<b>Design suggestions</b>	None
<b>Issues</b>	<p>The set of rights that will be available to the user for purchase is a subset of the original Potential Available Rights (PAR) that depends on the following factors:</p> <ul style="list-style-type: none"> <li>○ Original PAR;</li> <li>○ Rights acquired by the distributor (eventually already restricted either by the rights owner at selling time or by any other distributor that is preceding the present one in the sale value chain);</li> <li>○ Rights that the distributor decides to present the user as available (and that have to be less or equal to the acquired ones).</li> </ul> <p>Potential Available Rights (PAR) are defined by right owner who can then grant some rights combination to right purchaser. In case among acquired PAR there is the right to issue licenses or grant some rights (always a subset of the acquired ones), the new issuer can define the subset of PAR that will be available to the next actor in the chain. This brings to a nested set of restrictions that are supported by a chain of licenses each enforcing the correct set of rights and ensuring the possibility for the actor to properly use the object respecting the DRM chain. In the following example is sketched a simple sequence with a main owner, two chained distributors (for example EU and National level) and the user.</p>

#### 16.2.24 Content fruition

<b>UCId</b>	UC16.2.24
<b>Use case</b>	Content fruition
<b>Description</b>	The user accesses to selected content for fruition
<b>Actors</b>	User
<b>Assumptions</b>	All components are active and properly functioning
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user should log onto the Mobile Portal</li> <li>2 The user enters the special section where is stored the list of contents for which a license with some rights has been acquired.</li> <li>3 The user selects a content title to play it</li> <li>4 The AXMEDIS Player(s) present on the user device is activated</li> <li>5 The AXMEDIS player(s) performs all needed rights checking before playing</li> </ol>
<b>Post-conditions</b>	The user has been able to use the acquired content (and related fruition)
<b>Variations</b>	The same operation could be performed for a different business model (pay per view, pay per access, pay per download...
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	The PMS (which contacts AXCS) offers a functionality called “verify” which is used to check the user status in the system, verify the tool integrity and send the list of actions performed offline. This operation is performed before authorising the user or the first time the tool is used
<b>Issues</b>	None

#### 16.2.25 User Data Update

<b>UCId</b>	UC16.2.25
<b>Use case</b>	User Data Update
<b>Description</b>	A registered user modifies data previously inserted at registration time
<b>Actors</b>	User, mobile distribution portal and application
<b>Assumptions</b>	All components are active and properly functioning

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 A registered user logs onto the system</li> <li>2 The user requests to modify data previously inserted at registration time</li> <li>3 The Mobile Portal offers a specific section where the user is prompted with inserted data.</li> <li>4 The User can revise and update</li> <li>5 Once ended the user either confirm changes (or aborts)</li> <li>6 User new data are handled by the front end, which forwards request to the backend</li> <li>7 The backend retrieves user data from the database through User Management and User Profile modules</li> <li>8 Updated information is then forwarded and stored in the database.</li> </ol>
<b>Post-conditions</b>	User data have been successfully updated
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 17 AXMEDIS for Distribution towards i-TV

### 17.1 User Terminal Installation and Configuration

#### 17.1.1.1 PC+DVB Card Terminal

<b>UCId</b>	UC17.1.1.1
<b>Use case</b>	User Hardware Installation
<b>Description</b>	The user installs the required hardware in his PC
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user's PC is connected to a satellite dish, correctly pointed to the satellite providing the Data Broadcast. The user's PC has a PCI slot, an Ethernet port, or an USB connector free for installing the DVB-IP adapter.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user buys or obtains a DVB-IP satellite adapter suitable for his PC configuration (depending on operating system, available ports, etc.) and supported by the AXMEDIS Client Application</li> <li>2 The user physically installs the DVB-IP adapter according to the installation instructions provided by the manufacturer</li> <li>3 The user connects the DVB-IP adapter to the satellite dish</li> <li>4 The user boots the PC and installs any required software, driver or application, as specified by the manufacturer in the installation instructions, and in the AXMEDIS Client Application user manual</li> <li>5 The user configures the DVB-IP adapter according to instructions</li> <li>6 The user checks that the satellite signal is received correctly</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	Some DVB-IP cards might not support all AXMEDIS functionalities. It is recommended that the user obtains a supported DVB-IP adapter.
<b>Asynchronous actions</b>	Interactions with operating system components (e.g., firewall) or installed software (e.g., antivirus) could stop the DVB-IP adapter from working correctly. Installation of out-of-date drivers, or installation procedure not compliant with instructions, might stop the DVB-IP adapter from working correctly.
<b>Design suggestions</b>	A list of compatible adapters should be prepared. Full installation instructions should be given to the user.
<b>Issues</b>	If the satellite signal is not received correctly, there could be a problem in the pointing of the satellite dish, or in the satellite cable, or in the DVB-IP installation. Problems must be solved before proceeding. Occasional loss of signal (e.g., in presence of heavy rain or wind) does not represent a major problem; however, it may impact the fruition of service during and after the problem. It is recommended that the satellite dish installation be done by a professional.

#### 17.1.2 User Software Installation

<b>UCId</b>	UC17.1.12
<b>Use case</b>	User Software Installation
<b>Description</b>	The user installs the AXMEDIS Client Application
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The DVB-IP hardware is correctly installed and configured. The user PC has a working connection to the Internet.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user obtains the AXMEDIS Client Application Setup (e.g., from the Internet, from a CD, ...)</li> <li>2 The user runs the AXMEDIS Client Application Setup</li> <li>3 The user follows the steps of the installation</li> <li>4 Some information entered by the user is used to create his profile, that is securely communicated via Internet to a server</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	The exact form of the Setup, and the installation steps, could depend on the exact operating system of the user's PC, and on its configuration. Also, the user profile (country, language, gender ...) may influence some steps (e.g., subscription to language-specific services).
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	Different Setups applications could be distributed according to country, bundling with DVB-IP adapters, commercial promotions, etc. These Setups should contain minimal 'intelligence' and should be driven by the server, in order to make updates easier. The Server shall implement algorithms to detect the user language and provide him useful information on the service.
<b>Issues</b>	None.

### 17.1.3 User Registration

<b>UCId</b>	UC17.1.3
<b>Use case</b>	User Registration
<b>Description</b>	The user registers himself in order to access the AXMEDIS service
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has successfully installed the hardware and software AXMEDIS components.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user runs the AXMEDIS Client Application registration procedure</li> <li>2 The user obtains required authorizations (e.g., login/password) to access the AXMEDIS system. (This could require paying a subscription fee, a pre-paid amount, etc.)</li> <li>3 The AXMEDIS Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)</li> </ol>
<b>Post-conditions</b>	The user is ready to use the AXMEDIS service and access the published Content. (Access can be restricted only to some components)
<b>Variations</b>	The user may re-run the procedure to update authorizations. In some situations this procedure could be automatic and hidden to the user.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	The Server shall store flags about user registration.
<b>Issues</b>	None.

#### 17.1.3.1 Application Selection

Not mandatory in AXMEDIS

<b>UCId</b>	UC17.1.3.1
<b>Use case</b>	Application Selection

<b>Description</b>	The user selects the preferred Application model
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has successfully registered himself in order to fully access the AXMEDIS service.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user runs the AXMEDIS Client Application</li> <li>2 The user select the desired Application model between the three available: <ul style="list-style-type: none"> <li>o Standard application</li> <li>o Cache-based Distribution on i-TV</li> <li>o Cached-based Personalised Content Distribution</li> </ul> </li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	The user can change the selected Application with no restrictions
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	The Application selection should be easy and fast to perform
<b>Issues</b>	None

### 17.1.3.2 User Profiling

Not mandatory in AXMEDIS

<b>UCId</b>	UC17.1.3.2
<b>Use case</b>	User Profiling
<b>Description</b>	The user provides his/her preferences about AXMEDIS contents
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has successfully registered himself in order to fully access the AXMEDIS service.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user runs the AXMEDIS Client Application User Profiling procedure</li> <li>2 The user provides or updates his/her preferences about AXMEDIS contents</li> <li>3 The user decides if let the client application to automatically include the personal choices related to the AXMEDIS objects in the cache and his/her choices</li> <li>4 The user is aware of the profiling information that is sent back to the server and may decide to avoid the disclosure of personal information</li> <li>5 The user saves the his/her User Profile</li> <li>6 The user connects to the Internet to update his/her User Profile stored in the Server</li> </ol>
<b>Post-conditions</b>	None
<b>Variations</b>	The user may re-run the procedure to update his/her profile. The User profile is also automatically updated depending on the actual behaviour of the user (which content has been played, how many times it has been played, saved in the file system, etc.). When the user connects to the Internet, his/her user profile on the server is updated as well.
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 17.2 Content Listing

### 17.2.1 Content Web Listing

Not mandatory in AXMEDIS

<b>UCId</b>	UC17.2.1
<b>Use case</b>	Content Web Listing

<b>Description</b>	The user accesses a web page containing the list of the proposed AXMEDIS content in order to express his interest. He browses and previews the content listed in order to find the interesting content for him. He expresses his preference by voting. He can receive a given content either by push (if the content has received a lot of preferences) or by pull (if the content has not been selected for entering in the most voted top-list).
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user shall have an Internet Connection, needed to reach the web page where the proposed AXMEDIS Content Web List is published. The user shall know the URL where the Web List is published.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user reaches the AXMEDIS Content Web List</li> <li>2 The user displays the proposed content using different criteria (type, author, content producer, production date)</li> <li>3 The user inserts some key words for filtering Object potentially interesting for him</li> <li>4 The user reads all available information (contained in the AXMEDIS Info) associated to the AXMEDIS Object, helpful for voting</li> <li>5 The user plays, by downloading needed data, some short previews (if this option is available)</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step (Content Voting)
<b>Variations</b>	This AXMEDIS Content Web List could be published by another distributor (e.g., Tiscali, OD2, iLabs, Sejer, etc.). Eutelsat could synchronize his AXMEDIS library after getting the most voted top-list (this could optimise the use of the shared bandwidth in the Satellite Data Broadcast).
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	Possible previews related to the AXMEDIS Object should be simply extracted in order to provide a short preview to the user in the listing phase before voting.
<b>Issues</b>	None.

### 17.2.2 Content Carousel Listing

<b>UCId</b>	UC17.2.2
<b>Use case</b>	Content Carousel Listing
<b>Description</b>	The user consults the list from the AXMEDIS Client Application of the AXMEDIS Carousel currently in transmission. He accesses in any moment (by opening his AXMEDIS Client Application Interface) to the current carousel list proposed for all AXMEDIS users. He browses and previews the content listed in order to find the interesting content for him.
<b>Actors</b>	The Content Consumer (user).
<b>Assumptions</b>	The user knows the type of search that he wants to start (generic, advanced, personalized, pre-stored, etc.)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user uses some pre-defined functionalities to filter the content</li> <li>2 The user applies its own profile (locally stored) to the AXMEDIS offer to best match his interest in the offered content</li> <li>3 The user enters some key words in the content browsing</li> <li>4 The user reads all available information (contained in the AXMEDIS Info) associated to the AXMEDIS Object, helpful for selection</li> <li>5 The user plays some short previews (if this option is available) associated to the AXMEDIS Object, previously extracted from the AXMEDIS Info and added to the Electronic Program Guide (constantly transmitted to AXMEDIS users) of the AXMEDIS Service.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step (Content Selection)
<b>Variations</b>	None

<b>Asynchronous actions</b>	Some changes in the internal sequence of the transmitting carousel could alter the expected start date of the AXMEDIS Objects. A specific notification should be provided to users those voted for a given content (e.g., by sending an email). A general notification could be generally sent in multicast for all users, having an AXMEDIS Client currently listening, to warn about the variation of the expected dates.
<b>Design suggestions</b>	Some enriched content contained in the AXMEDIS Info (metadata) should be simply extracted to be presented in the Electronic Program Guide as an instrument to personalise the content browsing.
<b>Issues</b>	None.

### 17.3 Content Voting

Not mandatory in AXMEDIS

<b>UCId</b>	UC17.3
<b>Use case</b>	Content Voting
<b>Description</b>	The user expresses one or more preferences (depending on the number of preferences he can express on a daily/weekly/monthly basis) about some AXMEDIS Objects contained in the AXMEDIS Content Web List.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user still has available preferences to vote Content in the proposed web list.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user chooses one or more AXMEDIS Objects he wishes to receive by push inside the AXMEDIS Carousel</li> <li>2 The user sends his preferences to the server side</li> <li>3 The user receives a receipt about his vote expression</li> <li>4 The user receives a notification saying if his AXMEDIS voted Object has entered in the AXMEDIS Carousel</li> </ol>
<b>Post-conditions</b>	Most voted AXMEDIS Objects will be pushed directly to final users. Who has voted the content will receive it automatically. Others can manually select a given content from the carousel list from the AXMEDIS Client Application.
<b>Variations</b>	None.
<b>Asynchronous actions</b>	The voting action could affect the user profile. A synchronisation between the local stored and the server stored user profiles could be done.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

### 17.4 Content Selection

#### 17.4.1 Manual Content Selection

<b>UCId</b>	UC17.4.1
<b>Use case</b>	Manual Content Selection
<b>Description</b>	The user selects (manually) the scheduled content that will be received at the indicated time by push.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user leaves the computer and the AXMEDIS Client Application turned on during the time window of the selected transmission.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user double clicks on the AXMEDIS Object in order to select it for reception</li> <li>2 The user retrieves his selected AXMEDIS Object in the Downloading panel of the Client Application Interface. It means that the content has been scheduled for reception</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	The user could select his AXMEDIS Object from a remote computer and the order could be passed to his local AXMEDIS Client Application.

<b>Asynchronous actions</b>	The user could turn off the AXMEDIS Client Application (or the PC) and the reception could not be successful. Incompatible request done on a forbidden transponder.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

#### 17.4.2 Automatic Content Selection

<b>UCId</b>	UC17.4.2
<b>Use case</b>	Automatic Content Selection
<b>Description</b>	The user has voted for an AXMEDIS Object that has been added to the AXMEDIS Carousel. He receives automatically the voted content by push.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has turned on his AXMEDIS Client Application in the recommended time window to receive the Content correctly.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user receives a message notifying the expected start date of his previously voted AXMEDIS Object</li> <li>2 The user turns on his AXMEDIS Client before the transmission starts</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	The user can select automatically some other contents, typically system files or AXMEDIS Client Application updates.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	Design a solid environment where the AXMEDIS Client can be simply auto-updated.
<b>Issues</b>	None.

#### 17.5 Content Reception

<b>UCId</b>	UC17.5
<b>Use case</b>	Content Reception
<b>Description</b>	The user can check any time that the progress-bar, indicating the download state, is advancing.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has selected, either manually or automatically, an AXMEDIS Object distributed in the AXMEDIS Carousel.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the jobs panel where all current downloads are displayed</li> <li>2 The user reads the remaining time for the end of transmission</li> <li>3 The user can open the folder where the content is being received</li> <li>4 The user can interrupt the reception of a given content</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	The user, after opening the folder where the content is being received, deletes an incomplete and/or temporary file. This could put the AXMEDIS Client Application in an inconsistent state.
<b>Design suggestions</b>	None.
<b>Issues</b>	None.

#### 17.6 Content Reparation

<b>UCId</b>	UC17.6
<b>Use case</b>	Content Reparation
<b>Description</b>	The user receives the AXMEDIS Object, but in the access panel the demanded Content has a specific icon, indicating that the transmission is incomplete. The remaining lost packets can be downloaded in unicast by pull.

<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has an Internet Connection, needed to contact the server in order to repair the incomplete AXMEDIS Object.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user tries to open an AXMEDIS Object from the access panel provided from the Client Application Interface</li> <li>2 The user receives a pop-up saying that some packets were lost during the multicast transmission</li> <li>3 The user decides either to repair the Object via unicast or to wait for a next retransmission or to delete the incomplete Object.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	<p>The user receives a corrupted Object. No packets were lost but the checksum at the server side does not match with that one calculated at the client side. The user can try to open anyway the Object.</p> <p>The lost packets during the transmission of AXMEDIS Objects that have been explicitly requested by a user shall be notified to the server; the server shall decide the most effective recovery technique, either based on the retransmission of packets in unicast (via Internet) or redelivering the packets in the subsequent data carousel.</p>
<b>Asynchronous actions</b>	The user tries to repair an AXMEDIS Object too old. The AXMEDIS Object could be not more available for repairing.
<b>Design suggestions</b>	Definition of a strategy to combine Push/Pull technologies.
<b>Issues</b>	None.

## 17.7 Content Access

<b>UCId</b>	UC17.7
<b>Use case</b>	Content Access
<b>Description</b>	The user accesses his local cache containing several AXMEDIS Objects.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The AXMEDIS Content is successfully received.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user accesses the AXMEDIS Object for playing it</li> <li>2 The AXMEDIS Object is delivered to either the AXMEDIS Viewer or the standard application (with an additional AXMEDIS plug-in)</li> <li>3 The application detects if the Object needs to acquire a license</li> <li>4 The application finds a pre-acquired license for the Object and play it</li> <li>5 The application needs a new license for the Object and tries to contact the AXCS.</li>   <li>6 The user asks corresponding PMS for authorisation of operation</li> <li>7 The PMS sends authorisation result to the user, and keys for unprotecting them, if needed</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	If the user is not authorised because she has no license, then it contacts PMS for acquiring it.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

## 17.8 Content Preview

<b>UCId</b>	UC17.8
-------------	--------

<b>Use case</b>	Content Preview
<b>Description</b>	The user browses one/more AXMEDIS Object(s). The user opens and plays some short previews (if they are available) integrated in the received AXMEDIS Object. The user decides to buy or not the received AXMEDIS Content.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The AXMEDIS Object has been integrally received.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the AXMEDIS Object locally stored in his local cache</li> <li>2 The user browses the AXMEDIS Object, using the AXMEDIS Info associated to the Object</li> <li>3 The user reaches a preview available for the Object</li> <li>4 The user plays the AXMEDIS Object Preview</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	One or more previews (depending on the internal structure of the AXMEDIS Object) should be available for the final user, in order to help him in the content evaluation before purchasing it.
<b>Issues</b>	None.

### 17.9 License Acquisition

<b>UCId</b>	UC17.9
<b>Use case</b>	License Acquisition
<b>Description</b>	A user tries to acquire a license for consuming a protected and governed AXMEDIS Object.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	<p>The user should have an Internet connection (well functioning) in order to reach the AXCS and obtain the license for playing an AXMEDIS Object.</p> <p>The user belongs to the AXMEDIS community authorized to receive the AXMEDIS Content. The user has obtained required authorizations (e.g., login/password) to access the AXMEDIS system.</p> <p>The user station should support all secure protocols.</p>
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user wants to acquire an AXMEDIS object to play the protected resources within it</li> <li>2 The AXMEDIS Object is delivered to the application/viewer charged to open/play it</li> <li>3 As the AXMEDIS Object is protected and governed, the user is informed that he has to purchase the appropriate license for playing it</li> <li>4 The user requests the appropriate license</li> <li>5 The license server generates the license with the relevant parameters (principal, right/s, resource and conditions) and validates the license</li> <li>6 If the license is valid, the license server stores the license in the database of DRM licenses. If not, returns an alert with an explicative message.</li> <li>7 If the license has been generated and is valid, the license server returns to the actor the license ID or the license in clear-text or protected.</li> </ol>
<b>Post-conditions</b>	The user shall respect the rules contained in the received authorization.
<b>Variations</b>	<p>Silent license acquisition (contact a web site to have a license, after asking the user authorization). The License Acquisition could be for free in order to promote some special events.</p> <p>The acquiring computer could not be the same of the consuming computer (license could be acquired in a desktop, but the content could be played by a PDA)</p>

<b>Asynchronous actions</b>	The user could not be authorized to acquire licenses. The user can ask to enter in the AXMEDIS Community even after the reception of the AXMEDIS Object. Periodic system checks should be performed and in case of negative result system should be not more operational.
<b>Design suggestions</b>	None.
<b>Issues</b>	E-commerce backend and transactional functionalities should be available and in place. Security, privacy and transparency should be some fundamental basis.

### 17.9.1 User Identification

<b>UCId</b>	UC17.9.1
<b>Use case</b>	User Identification
<b>Description</b>	The user will be requested to identify and provide credentials needed to ensure that the requested transaction (purchase/rental) is valid and legal.
<b>Actors</b>	The Content Customer (user) (involved in the purchase/rental operation) The AXMEDIS Certifier (entity performing all required checks to ensure that purchase/rental operations are valid and legal)
<b>Assumptions</b>	See License Acquisition Use Case.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user enters his identification information (this does not necessarily mean personal details, it will be sufficient to have proper credentials, e.g., login/password)</li> <li>2 The user credentials are sent to the AXCS for verification</li> <li>3 The user waits for the server response</li> <li>4 If the user is identified as a regular one permission to proceed is granted, otherwise purchase procedure is aborted and user is sent back to browsing</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	See License Acquisition.
<b>Design suggestions</b>	None.
<b>Issues</b>	See License Acquisition.

### 17.9.2 Billing

<b>UCId</b>	UC17.9.2
<b>Use case</b>	Billing
<b>Description</b>	The user confirms the intention of purchasing the selected AXMEDIS Content. The user provides payment related information along with data needed to ensure legal validity of requested operation. The user accesses to the service on a prepaid subscription basis. If the user has enough credits to purchase the content the transaction is performed and the system can delivery the license for the AXMEDIS Object.
<b>Actors</b>	The Content Customer (user) (involved in the purchase/rental operation) The AXMEDIS Certifier (entity performing all required checks to ensure that purchase/rental operations are valid and legal)
<b>Assumptions</b>	See License Acquisition Use Case.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The AXCS shows to the user all billing information available including: <ul style="list-style-type: none"> <li>○ Price</li> <li>○ Conditions for each selected item</li> <li>○ Related use licence</li> <li>○ Scope and limitations</li> <li>○ Possible constraints</li> </ul> </li> <li>2 The AXCS asks the user to verify and accept presented terms</li> <li>3 If the user accepts procedure continues otherwise is aborted and user is sent back to browsing</li> <li>4 The user shall finalise billing information</li> <li>5 Once billing information are provided the user is requested to select the payment method (credit card, electronic wallet, pre paid card, pre assigned tokens or similar)</li> <li>6 The AXCS requires clearance to the AXMEDIS Distributor for the provided payment ID.</li> <li>7 If payment ID is cleared the user will be charged the cost</li> <li>8 The AXCS provides the system the proper clearance and the license delivery starts.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	A supplementary actor could be a bank or other institution that will handle the money transaction and has to be a third trusted party for both the user and the AXMEDIS Certifier.
<b>Asynchronous actions</b>	See License Acquisition
<b>Design suggestions</b>	None.
<b>Issues</b>	See License Acquisition.

## 17.10 Content Backup

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.10
<b>Use case</b>	Content Backup
<b>Description</b>	The user copies some interesting content in a backup support (either external or internal).
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	<p>The user can have some functionalities (API) that can make sure this operation of backup (the utility should ensure the integrity of the copied AXMEDIS Object, taking into account that an Object could be internally combined with other Objects).</p> <p>The Backup Operation has to be expressly authorized in the license terms.</p>
<b>Steps</b>	<ol style="list-style-type: none"> <li>6 The user opens the backup interface of the AXMEDIS Client Application</li> <li>7 The user selects all Objects involved in the backup operation</li> <li>8 The user specifies the backup unit where the AXMEDIS Content has to be copied.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	The AXMEDIS Content could be automatically deleted during the backup operation.
<b>Design suggestions</b>	The Backup operation should interact with the Intelligent Cache Manager before starting.
<b>Issues</b>	None.

## 17.11 Content Restore

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.11
<b>Use case</b>	Content Restore
<b>Description</b>	The user restores some previously backup AXMEDIS Object from the backup support (either external or internal).
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user can have some functionalities (API) that can make sure this operation of restore (the utility should ensure the integrity of the restored AXMEDIS Object, taking into account that an Object could be internally combined with other Objects). The Backup Operation has to be expressly authorized in the license terms.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user opens the restore interface of the AXMEDIS Client Application</li> <li>2 The user selects all Objects involved in the restore operation</li> <li>3 The user specifies the backup unit where the AXMEDIS Content from which the Content has to be restored.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None.
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	The Restore operation should interact with the Intelligent Cache Manager after restoring.
<b>Issues</b>	None.

#### 17.11.1 Cache Preloading

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.11.1
<b>Use case</b>	Cache Preloading
<b>Description</b>	The user activates the Cache loading (the first time the Application is activating and after a Cache Cleaning)
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has already set up the User Profile. The Cache is empty.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user switches on the AXMEDIS Client Application</li> <li>2 The user activates the Cache preloading functionality</li> <li>3 The user waits for the Cache to be filled</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 17.12 Cache Cleaning

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.12
<b>Use case</b>	Cache Cleaning
<b>Description</b>	The user emptiesthe local cache
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The Cache has some AXMEDIS objects
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user runs the AXMEDIS Client Set Up application</li> <li>2 The user activates the Empty Cache functionality</li> <li>3 The user may immediately asks for a new Cache Preloading</li> </ol>

<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

### 17.13 Cache-Based Personalised Content Distribution specific Use Cases

#### 17.13.1 Automatic Content Access Set Up

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.13.1
<b>Use case</b>	Automatic Content Access Set Up
<b>Description</b>	The user makes the Cache-Based Personalised Content Distribution Application set up.
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has payed a subscription (yearly, monthly or weekly). The user has already set up the User Profile and activated the Cache Preloading
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user switches on the AXMEDIS Client Application</li> <li>2 The user selects the Cache-Based Personalised Content Distribution Application</li> <li>3 The user waits for the AXMEDIS default Channels to be composed</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 17.13.2 AXMEDIS Channel personalisation

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.13.2
<b>Use case</b>	Content Access
<b>Description</b>	The user personalises an AXMEDIS Channel
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has already made the Automatic Content Access set up
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user switches on the AXMEDIS Client Application</li> <li>2 The user selects the Cache-Based Personalised Content Distribution Application</li> <li>3 The user changes the personal profile</li> <li>4 The user waits for the Cache preloading</li> <li>5 The user waits for the personalised AXMEDIS Channels to be composed according to the new user profile configuration</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 17.13.3 Automatic Content Access

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.13.3
<b>Use case</b>	Content Access
<b>Description</b>	The user plays an AXMEDIS Channel
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has already made the Automatic Content Access set up
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user switches on the AXMEDIS Client Application</li> <li>2 The user selects the Cache-Based Personalised Content Distribution Application</li> <li>3 The user selects an AXMEDIS channel</li> <li>4 The user plays the AXMEDIS channel</li> </ol>
<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	
<b>Issues</b>	None.

#### 17.13.4 AXMEDIS Channel PVR functionalities

Not mandatory in AXMEDIS

<b>UCId</b>	UC16.13.4
<b>Use case</b>	Content Access
<b>Description</b>	The user activates the PVR functionalities graphic interface
<b>Actors</b>	The Content Consumer (user)
<b>Assumptions</b>	The user has already made the Automatic Content Access set up
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user switches on the AXMEDIS Client Application</li> <li>2 The user selects the Cache-Based Personalised Content Distribution Application</li> </ol> <p>The user activates the PVR functionalities:</p> <ul style="list-style-type: none"> <li>• play</li> <li>• fast forward</li> <li>• rewind</li> <li>• record (the user records some content from an AXMEDIS channel to the hard disk)</li> <li>• pause</li> </ul>
<b>Post-conditions</b>	
<b>Variations</b>	
<b>Asynchronous actions</b>	None.
<b>Design suggestions</b>	The rewind functionality greater than a prefixed range of time (e.g. half an hour) will not be allowed (the content could not be present in the cache any more).
<b>Issues</b>	None.

## 18 AXMEDIS for Distribution to PDA via Kiosks

### 18.1 Content Catalogue Creation

<b>UCId</b>	UC18.1
<b>Use case</b>	Content Catalogue Creation
<b>Description</b>	The kiosk manager creates a catalogue
<b>Actors</b>	The kiosk manager, AXMEDIS Content Production
<b>Assumptions</b>	The kiosk manager is a registered AXMEDIS user with a specific UID and has all the right and tools to perform the operation (this operation is performed in the kiosk factory)
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The kiosk manager initiates catalogue preparation procedure (@kiosk factory)</li> <li>2 The kiosk manager inserts the following data in the kiosk factory catalogue creation user interface: <ol style="list-style-type: none"> <li>2.1 Catalogue file name (mandatory)</li> <li>2.2 Catalogue identifier (mandatory)</li> <li>2.3 Catalogue description</li> <li>2.4 Catalogue template</li> </ol> </li> <li>3 The kiosk manager performs a query with the query user interface to retrieve the list of object suitable for being acquired and reported in the kiosk content catalogue</li> <li>4 The query support system returns a list of AXOID and related metadata</li> <li>5 The Kiosk manager browse the list and identifies the needed objects accessing to public metadata and preview samples stored in AXINFO for each AXOID of the received list.</li> <li>6 The kiosk manager selects the desired objects to be included into the catalogue</li> <li>7 The kiosk manager adds additional information on a per item basis, for the time being it is expected to operate by adding allowed grants and ranking orders: <ol style="list-style-type: none"> <li>7.1 Selecting ranking for the Top 10 category</li> <li>7.2 Selecting ranking for the Best pick category</li> <li>7.3 Selecting ranking for the Offer category</li> </ol> <p>These operation are repeated up to catalogue completion, it is expected to perform it automatically in a second phase exploiting analysis of statistical reports on content/object usage</p> </li> <li>8 The kiosk manager adds required information on allowed grants specifying: <ol style="list-style-type: none"> <li>8.1 Rights available for purchase by the end user</li> <li>8.2 Validity of rights (from – to)</li> <li>8.3 Additional specification on allows use</li> <li>8.4 Country and Region of validity for the grant</li> <li>8.5 Fee type, amount and currency</li> <li>8.6 Back account for the fee reception</li> </ol> </li> <li>9 These operation are presently manually repeated up to catalogue completion, it is expected to perform it automatically in a second phase</li> </ol>
<b>Post-conditions</b>	The system shall have returned in operational mode
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Backend and transfer functionalities should be available and in place

### 18.2 Content Catalogue Loading (publication)

<b>UCId</b>	UC18.2
<b>Use case</b>	Content Catalogue Loading (publication)
<b>Description</b>	The kiosk manager publishes a catalogue
<b>Actors</b>	The kiosk manager, local AXDB, AXMEDIS Publication environments
<b>Assumptions</b>	The kiosk manager is a registered AXMEDIS user with a specific UID and has all the right and tools to perform the operation
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The kiosk manager initiates catalogue sending procedure (@kiosk factory)</li> <li>2 The kiosk manager inserts the following data in the kiosk factory catalogue transmission user interface: <ol style="list-style-type: none"> <li>2.a Catalogue file name (mandatory)</li> <li>2.b Catalogue target domain (mandatory)</li> </ol> </li> <li>3 The kiosk manager activates the sending procedure</li> </ol>
<b>Post-conditions</b>	The catalogue should be properly sent and finally received at target destination
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Backend and transfer functionalities should be available and in place

### 18.3 Content Catalogue Loading

<b>UCId</b>	UC18.3
<b>Use case</b>	Content Catalogue Loading
<b>Description</b>	The kiosk catalogue management module loads the new version of the catalogue either because a new one has been received or because the validity period has expired.
<b>Actors</b>	The kiosk catalogue management, local DB
<b>Assumptions</b>	The kiosk should be fully operational, catalogues should be changed on a scheduled basis with an expected change rate of 3-4 weeks.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The kiosk catalogue management module checks current catalogue validity and finds that a change is needed (either for reception of a newer one or for validity expiration)</li> <li>2 The kiosk catalogue management module performs a “switch to maintenance mode”</li> <li>3 The kiosk system exits normal operational and enters in maintenance mode.</li> <li>4 The kiosk catalogue management module Loads the new catalogue</li> <li>5 The kiosk catalogue management module performs a “switch to normal mode”</li> </ol>
<b>Post-conditions</b>	The system shall have returned in operational mode
<b>Variations</b>	The operation can be manually performed by a system administrator using the “maintenance” part of the kiosk application front end after logging in
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	Backend and transfer functionalities should be available and in place

### 18.4 User registration to kiosk

<b>UCId</b>	UC18.4
<b>Use case</b>	User registration to kiosk
<b>Description</b>	The system is fully functional and a non already registered user interacts with the system to register and access the kiosk application
<b>Actors</b>	The user, the kiosk application

<b>Assumptions</b>	The user is not already registered to the kiosk. The kiosk is connected to the backend and all components are well functioning.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The system is fully functional; the user is at a POP that is in idle mode and currently displays the login page.</li> <li>2 Being not registered the user presses the “register” button on the login page.</li> <li>3 The application front-end presents the user a registration form with the following data: <ul style="list-style-type: none"> <li>○ Login ID (mandatory)</li> <li>○ Password (mandatory)</li> <li>○ First Name (mandatory)</li> <li>○ Last Name (mandatory)</li> <li>○ e-mail (mandatory)</li> <li>○ Birth date</li> <li>○ Telephone</li> <li>○ Mobile phone</li> <li>○ VAT</li> <li>○ Address (base on the following fields): <ul style="list-style-type: none"> <li>▪ <ul style="list-style-type: none"> <li>▪ state, town, street, number and post-code</li> </ul> </li> </ul> </li> <li>○ Preferred payment method: <ul style="list-style-type: none"> <li>▪ pre-paid-cards, credit card...</li> </ul> </li> <li>○ Preferred device</li> <li>○ Notes</li> </ul> </li> <li>4 The user provides the required data and confirms.</li> <li>5 The kiosk local application server properly formats the data and send a request to the AXMEDIS Registration Service</li> <li>6 In case of success the AXMEDIS Registration Service sends back to the kiosk user final UID</li> <li>7 The kiosks retrieves the registration clearance, stores provided UID and sends the confirmation e-mail to the user specified account and grants user access to the kiosk application and services.</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	In the kiosk scenario the case of a user registering for the 1 <sup>st</sup> time has the major drawback that is not possible to provide the user with a direct access to his mail account to check the confirmation send back via mail. The usage of sms instead can be limited by environmental factors that are too risky to be left out.

## 18.5 User login

<b>UCId</b>	UC18.5
<b>Use case</b>	User login
<b>Description</b>	The system is fully functional and the user interacts with the system to access the kiosk application
<b>Actors</b>	The user, the kiosk application

<b>Assumptions</b>	As far as the system is concerned all should be as in previous case. As far as the user interaction is concerned this may happen as follows: The user interacts with system locally (this means operating from the kiosk screen...) or remotely (via PDA or mobile) in “ad hoc network” mode using own device browser. Furthermore the user can interact with the application front-end to eventually change the GUI language (this operation affects the whole language setting of the application, is always available and always brings back to the previously displayed page once ended) It is assumed that a user may access repeatedly to the kiosk and therefore is necessary to have a set of environmental settings stored as well as non volatile chart and acquired content list.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user interacts with the application front-end inserting the requested login data (user ID and password) and confirming</li> <li>2 Filled in data structure is sent back to the kiosk user management</li> <li>3 The kiosk user management checks user information locally</li> <li>4 The kiosk user management sends user data to the AXCS for verification (via AXCS web service interface)</li> <li>5 The AXCS checks received info</li> <li>6 The AXCS logs the registration event</li> <li>7 The AXCS sends back to the kiosk user management a ACK</li> <li>8 The kiosk user management confirms the login to the application front end</li> <li>9 The application front end grants access to available services: application front-end presents the user a page allowing to: <ol style="list-style-type: none"> <li>9.a Browse the catalogue</li> <li>9.b Modify own data</li> <li>9.c View or revise the current chart</li> <li>9.d Browse acquired content</li> <li>9.e Change GUI language</li> <li>9.f Catalogue management (available only for administrative users)</li> <li>9.g User management (available only for administrative users)</li> <li>9.h Logout</li> </ol> </li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	In the kiosk scenario if something happens and the user is forced to log on anew on the system but has not yet accessed to the confirmation mail is necessary to use locally stored data to grant access if the initial registration procedure has been successful. Therefore the system will have to keep track of this and behave as previously specified.

## 18.6 Content Browsing & Previewing

<b>UCId</b>	UC18.6
<b>Use case</b>	Content Browsing & Previewing
<b>Description</b>	The system is fully functional and end-users can browse and preview the content listed in order to select and buy AXMEDIS developed/delivered.
<b>Actors</b>	The user, the kiosk application
<b>Assumptions</b>	As in previous case

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The system presents the content list</li> <li>2 The end user browses the list</li> <li>3 The end user selects an item</li> <li>4 The end user asks for content preview</li> <li>5 Depending on content format a preview is presented as follows: <ol style="list-style-type: none"> <li>5.a Brief description for text</li> <li>5.b Thumbnail for images</li> <li>5.c X sec sample for Audio (X will depend on IPR rules)</li> <li>5.d X sec sample for Video (X will depend on IPR rules)</li> <li>5.e X sec sample for Animations (X will depend on IPR rules)</li> <li>5.f X sec sample for Multimedia (X will depend on IPR rules)</li> </ol> </li> <li>6 The end users decides next step between: <ol style="list-style-type: none"> <li>6.a Activate acquiring procedure</li> <li>6.b Returning to browsing</li> </ol> </li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

## 18.7 Content Selection and Chart Management

<b>UCId</b>	UC18.7
<b>Use case</b>	Content Selection And Chart Management
<b>Description</b>	The end user selects a content and either proceeds to check out or goes back browsing
<b>Actors</b>	The user, the kiosk application
<b>Assumptions</b>	As in previous case
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The end user selects a specific content for addition to the chart</li> <li>2 The user requests to proceed either to check out or to continue browsing</li> <li>3 Depending on previous step results the system enters one of the following to states: <ol style="list-style-type: none"> <li>3.a Check out procedure activation</li> <li>3.b Browsing &amp; previewing mode</li> </ol> </li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	In case of rental the chart can also be composed of a single item chart. Once the selection is operated the checkout procedure is automatically started in order to bring the user soon to fruition.
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

## 18.8 Check Out Procedure Initiation

<b>UCId</b>	UC18.8
<b>Use case</b>	Check Out Procedure Initiation
<b>Description</b>	In this phase the system prepared subsequent steps in order to ensure that checkout procedure can be either terminated with a successful content purchase or with a full restoring of previous state and the user can either continue browsing or abandon interaction with the kiosk
<b>Actors</b>	The user, the kiosk application
<b>Assumptions</b>	As in previous case

<b>Steps</b>	1 The system enters protected mode 2 A secure connection is established with the certification authority
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

## 18.9 Purchasing / Acquiring / Renting

<b>UCId</b>	UC18.9
<b>Use case</b>	Purchasing / Acquiring / Renting
<b>Description</b>	In this phase the customer confirms own will to purchase/acquire/rent selected content and is requested to provide payment related information along with data needed to ensure legal validity of requested operation. If provided data and payment ID are valid the transaction is performed and the system will start delivery process (corresponding to next phase)
<b>Actors</b>	Customer (involved in the purchase/rental operation), Certifier (entity performing all required checks to ensure that purchase/rental operation is valid and legal) and a bank or other institution that will handle the money transaction and has to be a third trusted party for both the customer and the certification authority.
<b>Assumptions</b>	As in previous case plus the availability of valid certifications for ensuring proper handling of money transaction. All operation will be performed passing through the kiosk backend that is properly interfaced with the AXEMDIS framework.
<b>Steps</b>	1 The system presents the customer billing information available (including price and conditions for each selected item, related use licence, scope and limitations, possible constraints...) 2 The system asks the customer to verify and accept presented terms 3 If the customer accepts procedure continues otherwise is aborted and customer is sent back to browsing 4 Once accepted purchase/acquisition/renting conditions, the customer is requested to finalise billing information 5 The customer shall finalise billing information 6 Once billing information are provided the customer is requested to select the payment method (credit card, electronic wallet, pre paid card or similar) 7 The customer is requested to provide a valid ID for payment (credit card, electronic wallet, pre paid card or similar) 8 The Certification authority requires clearance to the third trusted party for the provided payment ID. 9 The thirds trusted party should provide clearance on payment ID (if this fails operation is aborted) 10 If payment ID is cleared the customer will charged the cost (including the third trusted party commission for service) 11 Certification authority provides the system the proper clearance and the delivery process can start.
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

## 18.10 Repository Selection

<b>UCId</b>	UC18.10
<b>Use case</b>	Repository Selection
<b>Description</b>	The system determines whether content is available locally (inside the kiosk being part of the locally stored top ten content) or has to be downloaded from the kiosk server or the AXEPTool
<b>Actors</b>	The kiosk application, AXMEDIS Certification Authority and any remote AXMEDIS database belonging to the federation and reachable from the local system
<b>Assumptions</b>	As in previous case
<b>Steps</b>	1 The system checks each selected item for local / remote availability 2 In case of remote availability a secure channel is established and data cached locally
<b>Post-conditions</b>	The system shall have entered the next procedural step (in case of remotely available content, local cached content will have to be removed)
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.11 Destination Target Identification (Unique Id For Target – Wifi)

<b>UCId</b>	UC18.11
<b>Use case</b>	Destination Target Identification (Unique Id For Target – Wifi)
<b>Description</b>	The system has to identify the end delivery platform and, via a unique ID generate the required licence for fruition.
<b>Actors</b>	The local system and the PDA / mobile device
<b>Assumptions</b>	As in previous case.
<b>Steps</b>	1 The system identifies the end-user device and extracts a unique ID
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.12 Delivery Template Selection (Depending On Device)

<b>UCId</b>	UC18.12
<b>Use case</b>	Delivery Template Selection (Depending On Device)
<b>Description</b>	The system elects the proper delivery template for the end use device
<b>Actors</b>	The local system, the end user fruition device
<b>Assumptions</b>	As in previous case
<b>Steps</b>	1 The system identifies the class of delivery device The system selects the template to be used for delivery
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.13 Delivery Format Selection (Depending On Content)

<b>UCId</b>	UC18.13
<b>Use case</b>	Delivery Format Selection (Depending On Content)
<b>Description</b>	Once delivery fruition device has been selected along with delivery template the system has to select the best possible format to ensure highest possible fruition quality.
<b>Actors</b>	The kiosk application, the user fruition device
<b>Assumptions</b>	As in previous case
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 Based on the user device identification and delivery template the system selects the delivery format</li> <li>2 The system verifies if the availability of all components</li> <li>3 The system combines the required components of the delivery template to complete the delivery format</li> <li>4 The system starts the preliminary checks necessary to ensure proper delivery</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.14 Billing

<b>UCId</b>	UC18.14
<b>Use case</b>	Billing
<b>Description</b>	The system finalises the purchase / acquisition process and produces the billing related information (this step is performed in parallel to the delivery)
<b>Actors</b>	The kiosk application, the AXMEDIS certification authority and the user fruition device
<b>Assumptions</b>	As in previous case
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The system formalises the economic transaction into a proper bill</li> <li>2 The system sends the billing info to the end-user (according to provided billing info)</li> <li>3 The system sends the billing info to the AXMEDIS certification authority for the required subsequent processing steps</li> </ol>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.15 Data Delivery

<b>UCId</b>	UC18.15
<b>Use case</b>	Data Delivery
<b>Description</b>	The selected data is loaded onto the end-user device (this step is performed in parallel to the billing one)
<b>Actors</b>	The kiosk application, the user fruition device, the customer
<b>Assumptions</b>	As in previous case

<b>Steps</b>	1 The system requires the customer to initiate the content download 2 The customer selects the final storage target destination (if possible) 3 The customer activates the download procedure
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.16 Check Out Procedure Closure

<b>UCId</b>	UC18.16
<b>Use case</b>	Check Out Procedure Closure
<b>Description</b>	The check out procedure is closed either because content has been acquired or the procedure has been aborted (either by the customer or by the system)
<b>Actors</b>	The kiosk application
<b>Assumptions</b>	As in previous case
<b>Steps</b>	1 The system notifies the customer that the checkout procedure has been terminated 2 The secure connection with the certification authority is released 3 The system exits protected mode
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

### 18.17 Successful Delivery Check (Recovery In Case Of Failure)

<b>UCId</b>	UC18.17
<b>Use case</b>	Successful Delivery Check (Recovery In Case Of Failure)
<b>Description</b>	As the download process may require some time the system should verify that no problem occur during the delivery phase if so a recovery phase should be started including (in the worst case) the transaction cancelling and customer refund
<b>Actors</b>	The local system, the end user fruition device, the customer
<b>Assumptions</b>	As in previous case

<p><b>Steps</b></p>	<ol style="list-style-type: none"> <li>1 The local system should monitor the download process to ensure a smooth delivery</li> <li>2 The kiosk delivery module identifies the target device</li> <li>3 Depending on target device the Kiosk delivery module acts as follows:             <ol style="list-style-type: none"> <li>3.a The target device is a terminal (POP)                 <ol style="list-style-type: none"> <li>3.a.1 The Kiosk delivery module adapts the content to the fruition device (if necessary)</li> <li>3.a.2 The Kiosk delivery module returns to the application front end the info needed to retrieve the locally cached AXMEDIS object(s)</li> <li>3.a.3 The application front end loads a page to confirm delivery and grant access to the AXMEDIS object(s)</li> </ol> </li> <li>3.b The target device is a user PDA                 <ol style="list-style-type: none"> <li>3.b.1 Adapts the content to the fruition device (if necessary)</li> <li>3.b.2 The Kiosk delivery module retrieves device data (kind, storage, certificate...)</li> <li>3.b.3 The Kiosk delivery module performs required check on received device data</li> <li>3.b.4 If checks are positive the Kiosk delivery module loads a page to ask downlad activation</li> <li>3.b.5 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)</li> <li>3.b.6 The kiosk delivery module takes the cached content from the local storage</li> <li>3.b.7 The kiosk delivery module retrieves from the local storage the kind of operation requested on the AXMEDIS object. If the requested operation is a purchase acts as follows:</li> </ol> </li> </ol> </li> </ol>
---------------------	---

<b>Steps</b>	<p>3.b.7.1 If the AXMEDIS object is a NOT governed one the kiosk delivery module requires the Local License generator to generate a “device based” license</p> <p>3.b.7.2 The Local License generator generates a “device based” license</p> <p>3.b.7.3 The Local License generator returns the kiosk delivery module the generated “device based” license</p> <p>3.b.7.4 Kiosk delivery module requires the AXCS to generate the due keys</p> <p>3.b.7.5 The AXCS retrieves the due keys</p> <p>3.b.7.6 The AXCS returns the kiosk delivery module the retrieved due keys</p> <p>3.b.8 The kiosk delivery module loads data onto the PDA (AXMEDIS object, keys and license for not governed objects)</p> <p>3.b.9 The kiosk delivery module monitors the download</p> <p>4 The kiosk delivery module notifies the Kiosk Application front end of successful closure of the check out procedure</p> <p>5 The user can now use the content according to acquired rights via the AXMEDIS viewer, while in case of problems the system should perform at least 3 retries</p> <p>6 Inform the customer of the incurred problem</p> <p>7 Ask the customer which choice is preferred among:</p> <p>7.a New set of delivery retry</p> <p>7.b Deferred delivery</p> <p>7.c Delivery cancel</p> <p>8 The system should take note of customer decision and consequently proceed to:</p> <p>8.a Activate a new set of delivery retry (maximum 3)</p> <p>8.b Deferred delivery</p> <p>8.b.1 Ask the customer the time of next delivery</p> <p>8.b.2 Schedule next delivery</p> <p>8.b.3 Flag the process for possible cancellation &amp; refund</p> <p>8.c Delivery cancel</p> <p>8.c.1 Enter secure mode</p> <p>8.c.2 Establish a secure connection with the AXMEDIS certification authority</p> <p>8.c.3 Performs a roll back request (including billing cancelling and money refund)</p> <p>8.c.4 The system notifies the customer that the delivery and related transaction has been annulated</p> <p>8.c.5 The system notifies the customer that refund procedure has been activated</p> <p>8.c.6 The secure connection with the certification authority is released</p> <p>8.c.7 The system exits protected mode</p> <p>9 The system goes back to normal operation mode allowing the customer to browse and select content</p>
<b>Post-conditions</b>	The system shall have entered the next procedural step
<b>Variations</b>	None
<b>Asynchronous actions</b>	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
<b>Design suggestions</b>	None
<b>Issues</b>	As in previous case

**18.18 Content fruition after download on PDA or Mobile**

<b>UCId</b>	UC18.18
<b>Use case</b>	Content fruition after download
<b>Description</b>	This step has to be performed every time the content is accessed in order to ensure proper fruition and full respect of DRM rules and constraints
<b>Actors</b>	The kiosk application, the AXOM, the domain PMS, the AXMEDIS Certification Supervisor, the user and the fruition device
<b>Assumptions</b>	The device has enough computational power, the connectivity among all involved actors is granted and stable, required elaboration time per request is below a reasonable threshold to ensure user acceptance
<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user requests access to the downloaded content</li> <li>2 The local viewer gets the license from the governed object</li> <li>3 The local viewer gets the AXOID from the governed object</li> <li>4 The local viewer gets the UID</li> <li>5 The local viewer gets the device ID</li> <li>6 The local viewer requires the PMS domain (via AXOM) the consistency of the required operation for the specified AXOID by the UID on the specific device with the given licence</li> <li>7 The viewer informs the user of being performing a licensing check and enters a wait state for either the keys or a NACK</li> <li>8 The domain PMS requires to the AXMEDIS Certification Supervisor to perform the check and if positive generate the related user keys</li> <li>9 The domain PMS waits for the either the keys or a NACK</li> <li>10 The AXMEDIS Certification Supervisor performs a license check on the basis of the requested usage, identified object, device and UID and decides whether the operation is feasible or not. According to check results it either: <ol style="list-style-type: none"> <li>10.a Sends back to the requesting PMS domain needed user keys (in case of positive result)</li> <li>10.b Sends back to the requesting PMS domain a NACK</li> </ol> </li> <li>11 The PMS domain receives the reply and forwards it to the requesting viewer (via AXOM)</li> <li>12 Depending on check results the viewer proceeds as follows: <ol style="list-style-type: none"> <li>12.a Allows content fruition</li> <li>12.b Blocks content fruition</li> </ol> </li> </ol>
<b>Post-conditions</b>	The user is free to perform further requests on the object, return to browse the catalogue, acquire new objects and / or leave the system.
<b>Variations</b>	none
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	If the object is purchased for personal use even once out of the Kiosk infrastructure the purchased license should be a device based one and the control should be able to find on the same device the following data: license, key and control data ensuring that license and key are related to the specifi end user device only

**18.19 User Profile Change**

<b>UCId</b>	UC18.19
<b>Use case</b>	User Profile Change
<b>Description</b>	The end user is logged-on and willing to change own profile.
<b>Actors</b>	The end user and the Kiosk Manager
<b>Assumptions</b>	The kiosk application is up and running, the back end is connected and functional.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The application front end has granted access to available services including:             <ol style="list-style-type: none"> <li>1.a Browse the catalogue</li> <li>1.b Modify own data</li> <li>1.c View support information</li> <li>1.d Logout</li> </ol> </li> <li>2 The user selects in the Content List the option “Modify own data”</li> <li>3 The system presents the user the profile form with the following data:             <ul style="list-style-type: none"> <li>○ Login ID (mandatory)</li> <li>○ Password (mandatory)</li> <li>○ First Name (mandatory)</li> <li>○ Last Name (mandatory)</li> <li>○ e-mail (mandatory)</li> <li>○ Birth date</li> <li>○ Telephone</li> <li>○ Mobile phone</li> <li>○ VAT</li> <li>○ Address (base on the following fields):                 <ul style="list-style-type: none"> <li>▪                     <ul style="list-style-type: none"> <li>▪ state, town, street, number and post-code</li> </ul> </li> </ul> </li> <li>○ Preferred payment method:                 <ul style="list-style-type: none"> <li>▪ pre-paid-cards, credit card...</li> </ul> </li> <li>○ Preferred device</li> <li>○ Notes</li> </ul> </li> <li>4 The user provides the required data</li> <li>5 The user confirms input operation ending either pressing a button on the interface or any other widget.</li> </ol>
<b>Expected results</b>	<ol style="list-style-type: none"> <li>1. The user should be registered</li> <li>2. The user should be assigned an AXMEDIS UID</li> <li>3. The system should be notified of the registration (via mail/sms)</li> <li>4. The user should be logged into the system</li> </ol>
<b>Post-conditions</b>	The user profile is successfully updated
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 18.20 User Device Configuration & Application Front-end Installation

<b>UCId</b>	UC18.20
<b>Use case</b>	User Device Configuration & Application Front-end Installation
<b>Description</b>	The end user configures own device (PDA...)
<b>Actors</b>	The end user, the Kiosk Front end application, the user fruition device (PDA...)
<b>Assumptions</b>	The kiosk front-end application is fully functional.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The user has access to a page with the following info:             <ol style="list-style-type: none"> <li>1.a How to connect the PDA / Tablet to the kiosk via WiFi (including how to test the connection)</li> <li>1.b How to download the Application client on the device (including how to test the client)</li> </ol> </li> <li>2 The user performs on the device the required operation to configure the WiFi connection</li> <li>3 The user performs the suggested check to ensure that WiFi configuration is successful</li> <li>4 Device connects to the kiosk application front end</li> <li>5 The application front end returns a test display page</li> <li>6 The user performs on the device the required operation to download the application client (following a specific URL returned in the previously provided test page)</li> <li>7 The device downloads the application client</li> <li>8 The user install the downloaded client</li> <li>9 The user performs the suggested check to ensure that application client install is successful</li> <li>10 Device connects to the kiosk application front end</li> <li>11 The application front end returns a test display object and a link to bookmark for future access via device</li> <li>12 The application client displays the test object</li> <li>13 The application client bookmarks the provided URL to access via device</li> <li>14 The installed AXMEDIS client connects to the domain PMS to perform the requested “Registration” &amp; “Authentication” as described in overall scenarios</li> </ol>
<b>Post-conditions</b>	The end user device is properly configured
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

### 18.21 Content Update (via Satellite)

<b>UCId</b>	UC18.21
<b>Use case</b>	Content Update (via Satellite)
<b>Description</b>	The kiosk content is updated via satellite
<b>Actors</b>	The kiosk manager, the Kiosk and the AXMEDIS framework
<b>Assumptions</b>	The kiosk and all its application are fully functional.

<b>Steps</b>	<ol style="list-style-type: none"> <li>1 The checking time is over a Down-Link channel check has to be performed</li> <li>2 The AXMEDIS B2B Satellite Reception Listener checks for data availability and behaves as follows:             <ol style="list-style-type: none"> <li>2.a Data is not available yet so a further check is scheduled and the application enters wait mode (cycling back to point 1)</li> <li>2.b Data is available therefore is downloaded (2.b.1) and progressively cached locally (2.b.2)</li> <li>2.c Received data is stored locally</li> </ol> </li> <li>3 The AXMEDIS B2B Satellite Reception Listener activates the AXMEDIS Action Manager to decide how to proceed</li> <li>4 The AXMEDIS Action Manager invokes the AXMEDIS B2B Satellite Reception Content Checker to verify consistency check on received data</li> <li>5 The AXMEDIS B2B Satellite Reception Content Checker proceeds as follows             <ol style="list-style-type: none"> <li>5.a Performs consistency check on received data</li> <li>5.b If result is positive returns ACK and control to the AXMEDIS Action Manager</li> <li>5.c If result is negative requires the distribution server to resend the damaged packages via Up-link as detailed here after:                 <ol style="list-style-type: none"> <li>5.c.1 Satellite Reception Content Checker requires missing or damaged packages via Up-Link</li> <li>5.c.2 Satellite Reception Content Checker receives missing or damaged packages via Up-Link</li> <li>5.c.3 Satellite Reception Content Checker returns ACK and control to the AXMEDIS Action Manager</li> </ol> </li> </ol> </li> <li>6 The AXMEDIS Action Manager retrieves the data from the local storage</li> <li>7 The AXMEDIS Action Manager extracts the content form the OpenSky package</li> <li>8 The AXMEDIS Action Manager checks the received data to determine what it is and behaves consequently:             <ol style="list-style-type: none"> <li>8.a Received data are AXMEDIS Object: data is stored in the AXDB</li> <li>8.b Received data are system / application updates: invoke the kiosk data manager to store data locally according to needs                 <ol style="list-style-type: none"> <li>8.b.1 The kiosk data manager stores the received data locally in plain format</li> </ol> </li> </ol> </li> </ol>
<b>Post-conditions</b>	The kiosk content is properly updated
<b>Variations</b>	None
<b>Asynchronous actions</b>	None
<b>Design suggestions</b>	None
<b>Issues</b>	None

## 19 Composite Use Case: automatic content production

Massive production of content, acquisition of content form a set of database via crawler and preparation of them for the distribution on distribution portal. The system will be ready to create licenses on the basis of a specific model defined by the distributor according to some different models.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
producer	configure crawling			searchbox control panel
producer	start crawling			searchbox control panel
producer	definition of acquisition rule	AXMEDIS rule		AXMEDIS Rule Editor
producer	run rule from the rule editor	AXMEDIS object		AXMEDIS Rule Editor
producer	make query on the database to find the object produced	list of AXMEDIS object		AXMEDIS Editor
producer	open the object from the AXDB and inspect the object			AXMEDIS Editor
producer	define to run automatically the rule when something is updated on the CMS			searchbox control panel
producer	make a change on the CMS information			CMS
producer	make query on the AXDB to search for updated content			AXMEDIS Editor
producer	inspect the updated object			AXMEDIS Editor

## 20 Composite Use Case: content editing, protection and authoring

A producer wants to produce a new AXMEDIS object getting raw content from the AXDB. She uses the internal/external editors to make changes to the raw content, uses the SMIL editor to produce a multimedia presentation, defines metadata (title, creator, etc.) as well as the DRM /protection information for the composed object.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
producer	create a new AXMEDIS object	AXMEDIS object		AXMEDIS Editor
producer	search for raw resources from database	list of AXMEDIS objects		AXMEDIS Editor
producer	integrate the basic resources into the object	composed AXMEDIS object		AXMEDIS Editor

producer	edit a basic resource using an internal editor/viewer			AXMEDIS Editor
producer	edit a basic resource using an external editor			
producer	produce a SMIL resource using the Visual Editor/Viewer	a SMIL resource		AXMEDIS Editor
producer	define the protection information using the Protection Editor/Viewer	protection information		AXMEDIS Editor
producer	define the PAR using the DRM Editor/Viewer	DRM information		AXMEDIS Editor
producer	add Metadata using the Metadata Editor	Metadata		AXMEDIS Editor
producer	upload the object on the AXDB			AXMEDIS Editor

## 21 Composite Use Case: content storage and search, database management

Content is produced, shared in the P2P network and then is searched, loaded in an AXMEDIS tool for editing, saved on the AXDB integrated with other contents. New content is then published again on the P2P network.

Who (producer, integrator, distributor, author, etc.)	Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.)	What is produced (resource, object, metadata, license, protection information, etc..)	Description of Content	AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)
Producer	creates a content (see Creation of a new AXMEDIS object )	AXMEDIS object	---	AXMEDIS Editor
Producer	shares a content on P2P network (see Manual Publication of AXMEDIS Objects with the AXEPTool )	AXMEDIS object	---	AXEPTOOL
Integrator	searches a content in the P2P network or in the AXDB (see Querying for AXMEDIS objects and inside objects)	Selection	---	Query Support, Query User Interface, AXEPTOOL, AXDB
Integrator	retrieves the desired contents (see View/Manage query results coming from the AXEPTool )	AXMEDIS objects	objects interesting for the integrator	Query Support, Selection Archive, AXDB
Integrator	modifies/integrates the objects (see Creation of a new AXMEDIS object)	AXMEDIS object	integrated object	AXMEDIS Editor
Integrator	saves the object (see Administer Objects in the AXMEDIS DB )	AXMEDIS object	newly created object	AXDB, Loader/Saver

Integrator	publish the object in the P2P network (see Manual Publication of AXMEDIS Objects with the AXEPTool)	AXMEDIS object	---	AXEPTOOL
------------	---	----------------	-----	----------

## 22 Composite Use Case: content acquisition for the domain and usage in the domain

A user purchases a license to use AXMEDIS content. The license allows to use the content in a domain. The user downloads the content together with the license. The user registers in the domain and tries to perform an action over the content using an AXMEDIS tool.

Who (producer, integrator, distributor, author, etc.)	Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES	What is produced (resource, object, metadata, license, protection information, etc..)	Description of Content	AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)
User	Purchase the license			
Distributor	Generate the license (see UC22.5.2.1 “License creation for new content”)	AXMEDIS license	Domain license	PMS Server
User	Download license and/or content			
User	Register in a domain (use case to be defined)	Info updated in domain manager	User belongs to a domain	AXMEDIS Domain Manager
User	Certify an AXMEDIS tool (first use of the tool) (see UC22.2.2.2” Certification of AXMEDIS tool by a user on a Device”)	Tool certificate, private key, enabling code, AXTID, AXCS database entry	After certifying AXMEDIS tool is ready to be used in the system.	AXMEDIS Tool
User	Perform an action over the resource (see UC22.5.2.5, UC22.5.2.6, UC22.5.2.7 “User authorisation based on licenses”)	Authorisation result. Storage of protection info in secure cache if positive authorisation		PMS Client PMS Server PMS Domain Home / Factory

## 23 Composite Use Case: content posting in the P2P and search from the P2P

A producer and/or integrator are interested in automating the publishing of objects and their download from the network, when they change. In addition, in the loading from the network, the application of some metadata mapping before loading them into the database may be needed. These activities are automated by the AXCP Engine in collaboration with the AXEPTool.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
Producer, integrator	Publish an object in the P2P network	AXMEDIS object	AXMEDIS object	AXCP Engine and AXEPTool
Integrator or distributor	Make a query on the query support for looking for objects into the P2P network	A Selection	AXMEDIS object, XML	Query Support and AXEPTool
Producer, integrator	Decide to permanently publishing an object in the P2P network	AXMEDIS object	AXMEDIS object	A tool to create a rule for publishing and making this active AXCP Engine and AXEPTool
Integrator or distributor	Decide to permanently download an object from the network	A Selection	AXMEDIS object, XML	A suitable user interface for the AXEPTool
Integrator or distributor	Decide to permanently load an object from the IN AXDB and from the network	A Selection and a rule	AXMEDIS object, XML	A tool to create a rule for loading and making them active AXCP Engine and AXEPTool

## 24 Composite Use Case: programme production and publication

This section describes the combination of the use case to create, edit a P&P programme using the Use Cases (10.x). These composite use cases require the use of the Query Support User Interface (UC 3.1.2) to add of AXMEDIS objects or selections. A programme manager can also set editing and request activation using the AXMEDIS Workflow where editing request and activation requests are made to the P&P Editor and the P&P Engine.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
Programme manager	UC10.1: Create a P&P Programme (P&P ID automatically generated)	New P&P Programme (XML)		P&P Editor
Programme manager	UC3.1.2: Use the Query Support User interface to select AXMEDIS objects or Selections	List of AXOIDS		P&P Editor with the integrated Query Support User interface (Web Interface)

Programme manager	UC10.2: Edit a selected P&P Programme	P&P Programme with Header, Schedule and Distribution parameters		P&P Editor
Programme manager	UC10.5: Trial pre-activation of a programme.	A response (success/failure) from the P&P Engine		P&P Editor and P&P Engine
Programme manager	UC10.3: Activate a selected P&P Programme	A response (success/failure) from the P&P Engine  Notification to WF.		P&P Editor and P&P Engine
P&P Engine	UC10.4: Launch a programme	Request to AXCP for content processing if necessary; Retrieval of the object to distribute to the specified distribution server  Command and Reporting to WF.		P&P Engine and the specified Distribution Server
Programme manager	UC10.6: P&P Engine Monitor	GUI visualisation of a list of currently active P&P Programme		P&P Engine and P&P Engine Monitor

## 25 Composite Use Case: workflow process definition

This section describes the usage for AXMEDIS workflow tools in terms of the use cases defined in section 7. By using the AXMEDIS workflow, the users can create processes, which when executed produces the required results. These processes is consistent of various activities that involve interaction with various AXMEDIS tools classified in four types viz. Object Editors, Engines, Rule Editors, Query Support. These processes are written using various workflow functions implemented such that these utilise various AXMEDIS tools. Therefore the execution of these processes in fact results in a logical and timely activation, de-activation and coordination of various AXMEDIS tools, which when enacted produces the desired results. E.g. consider the following steps of execution:

1. The user requests the creation of a new object using the AXMEDIS Object Editor
2. The Editor Notifies the workflow of the completion of the creation of the object
3. The user requests the creation of a new rule for formatting of the object using Rule Editor
4. The rule editor notifies the workflow of the completion of the creation of the rule
5. The user installs and activates the rule in the AXCP engine
6. The user requests the running of the rule in the AXCP engine
7. The AXCP engine executes the rule
8. The desired formatted object is produced

The above steps can form a typical workflow process, where each step is a single activity to be performed. So when the complete process is executed, a new object is created along with a new rule and a formatted version of the same object.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN TH E PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
Workflow Manager	Creates a new workflow process UC7.2.1.1, UC7.2.1.2, UC7.2.1.3, UC7.2.1.4, UC7.2.1.5, UC7.2.1.7	Workflow Process	A logical sequence of execution, which when executed produces the required results	AXMEDIS Workflow
Producer, User	Create AXCP Rules using an already defined workflow process (as above). UC7.2.1.9, UC7.2.1.10, UC7.2.1.11, UC7.2.1.12, UC7.2.1.14	AXCP Rule		Rule Editors,
Program Producer, Manager	Create PnP Program using an already defined workflow process (as above). UC7.2.1.9, UC7.2.1.10, UC7.2.1.11, UC7.2.1.12, UC7.2.1.14	PnP Program	XML P&P Programme which describes the distribution specification for a group or each individual AXMEDIS objects saved in the P&P Repository and held by WF	Programme and Publication Editor
Producer, User	Create a new AXMEDIS Object or edit an existing object. UC7.2.1.9, UC7.2.1.10, UC7.2.1.11, UC7.2.1.12, UC7.2.1.14	AXMEDIS Object, DRM, METADATA, Protection, SMIL, etc information		AXMEDIS Editors
Producer, User	Execute AXCP Rules using an already defined workflow process (as above). UC7.2.1.9, UC7.2.1.10, UC7.2.1.11, UC7.2.1.12, UC7.2.1.14	AXMEDIS Objects, license, protection, adaptation, formatting, etc		AXCP Engine
Producer, User	Search for Object in AXDB or P2P network. UC7.2.1.9, UC7.2.1.10, UC7.2.1.11, UC7.2.1.12, UC7.2.1.14	Selection	List containing AXOIDs of interest	AXDB Query Support
Producer, User	Load or Save Objects in AXDB. UC7.2.1.17, UC7.2.1.18	AXMEDIS Object	Newly created Object	AXDB Query Support

## 26 Composite Use Case: Content behaviour definition and usage on the client side

An author wants to produce a new SMIL resource based on available media resources from the AXDB. The SMIL editor (including visual editor, behaviour edit) is used to produce the visual and temporal information of the composed object. Object editor is used to produce the annotations and comment for the SMIL file and Internal SMIL player can be called to preview the final result of SMIL resource.

Who (producer, integrator, distributor, author, etc.)	Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES	What is produced (resource, object, metadata, license, protection information, etc..)	Description of Content	AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)
Author	Create composite multimedia scenes in two dimensional canvas	A new SMIL file or a new version of an existing file	Rectangular shapes placement in the canvas	Visual editor and viewer
Author	Link shape with audiovisual resources	New resources associated with the SMIL file	Shapes linkage with media resources	Visual editor and viewer
Author	Adjust these resources along timeline for displaying	Resources with temporal information	Temporal management along the timeline	Behaviour editor and viewer
Author	Add the comments and descriptions for	Resources with annotations and comments	Annotations and comment editing with text menu	Object editor and viewer
Author	Preview the SMIL resource		SMIL rendering	Internal SMIL player

## 27 Composite Use Case: production and execution of content processing rules

This use case describes main steps to execute for producing and put in execution content processing rules. The first tool to be used is the AXCP Rule Editor. It is used to edit a new or an existing AXCP Rule by defining properties in terms of firing conditions when it is active in the AXCP Rule Engine, parameters involved in the script, dependencies related to AXCP Tools to be used, the source code of script written using the AXCP Rule Language. If the production is dependent by AXMEDIS Workflow request, a notification is produced and sent back to communicate the end of the activity. When a rule is activated, it is transferred into the AXCP Rule Engine. The AXCP Rule Scheduler periodically tests the firing condition and when it is verified the rule is put in execution on a AXCP Rule Executor.

Who (producer, integrator, distributor, author, etc.)	Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES	What is produced (resource, object, metadata, license, protection information, etc..)	Description of Content	AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)
Producer	Create a new AXCP Rule	A new empty AXCP Rule		AXCP Rule Editor, AXMEDIS Workflow manager
Producer	Load an AXCP Rule	An AXCP rule		AXCP Rule Editor
Producer	Edit the rule	Schedule, the script of rule, rule parameters and dependencies		AXCP Rule Editor
Producer	Debug the script	Corrections, simulation, analysis		AXCP Rule Editor
Producer	Activate the AXCP Rule	A feedback notification from the AXCP Rule Engine		AXCP Rule Editor, AXCP Rule Engine
Producer	Send completion of activity	A workflow notification		AXCP Rule Editor, AXMEDIS Workflow manager
AXCP Rule Engine	Run an AXCP Rule	AXMEDIS Objects, license, protection, adaptation, formatting		AXCP Rule Scheduler and Executor, AXMEDIS AXCP Tools (plugins), Main Query Support, AXDB
AXCP Rule Engine	Send completion of activity	A workflow notification		AXCP Rule Scheduler and Executor, AXMEDIS Workflow manager

## 28 Composite Use Case: Content production and usage

Content is created at the content factory and then delivered to distributors (both for kiosk and mobiles) that finalise the process prior to perform actual distribution to users. Users acquire content and use it either at kiosks or on their mobile

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN TH EPREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
Producer	Creates an object (UC4.1.1)	AXMEDIS Object	HTML page (image & text)	AXMEDIS Editor
Producer	Protects an object and issues a license (UC13.3.1 and UC13.5.2.1)	License	License Protection info	AXMEDIS License editor Protection tool
Producer	Publishes an object for distribution (UC11.3, UC11.4 / UC10.2.2, UC10.2.5)	AXMEDIS protected Object	HTML page License Protection info	AXMEDIS P&P / AXEXPTool
Distributor	Retrieves an object (UC9.2.1 / UC10.2.9, UC10.2.10, UC10.2.13)	AXMEDIS protected Object	HTML page License Protection info	AXMEDIS Query Support
Distributor	Finalises distribution - adding pricing, use restrictions... - (UC)	AXMEDIS protected Object	HTML page License Protection info	AXMEDIS License editor Protection tool AXCP
Distributor	Distributes an object via kiosk (UC18.1-3/18.19-21)	AXMEDIS protected Object	HTML page License Protection info	Kiosk application AXMEDIS Framework
Distributor	Distributes an object via mobile (UC16.2.1-12)	AXMEDIS protected Object	HTML page License Protection info	Mobile application AXMEDIS Framework
User	Selects and acquires an object via kiosk (UC18.4-17)	AXMEDIS protected Object	HTML page License Protection info	Kiosk application AXMEDIS Framework
User	Uses the acquired object via kiosk (UC18.18)	AXMEDIS protected Object	HTML page License Protection info	Kiosk application AXMEDIS Framework
User	Selects and acquires an object via mobile (UC16.2.13-23)	AXMEDIS protected Object	HTML page License Protection info	Mobile application AXMEDIS Framework
User	Uses the acquired object via mobile (UC16.2.24)	AXMEDIS protected Object	HTML page License Protection info	Mobile application AXMEDIS Framework

## 29 Composite Use Case: contract reading and digital license production and usage

This composite use case describes how a contract is read to generate an associated license or PAR for an AXMEDIS object.

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.) REFER TO USE CASES DEFINED IN THE PREVIOUS PAGES</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
User	Select contract to be analysed and pass it to the corresponding tool	License (incomplete) or PAR	The clauses and information extracted from the contract is inserted in the license. Other information needed will be extracted as well	Contract clauses extractor (to be done)
User	Finalise license with contract clauses	License (complete) or PAR	The user inserts information that was not recognised from contract, but information was there	DRM Editor and viewer
User	Store license or PAR			DRM Editor and viewer

### 30 Composite Use Case: Access to action logs and their usage

An end user uses an object, so that an event reporting is generated. This log is then collected by AXMEDIS tools such as AXCS and gathered by CAMART until it is integrated in the different business user CMSes by the Administrative Information Integrator

<b>Who (producer, integrator, distributor, author, etc.)</b>	<b>Action Performed (create, assign id, certify, authenticate, register, make query, open, access, load, save, extract, copy, play, move, send, etc.)</b>	<b>What is produced (resource, object, metadata, license, protection information, etc..)</b>	<b>Description of Content</b>	<b>AXMEDIS Tool Name or NAXT (not with AXMEDIS tool)</b>
end user	perform an action on an object (see 3.1.10) (see Viewing/Using of AXMEDIS objects (DSI: Bellini))	information on usage	---	AXMEDIS player
distributor	retrieve administrative logs (see 13.2.6.4) (see Usage report about a distributor)	information on usage	---	AII and CAMART
author	retrieve administrative logs (see 13.2.6.3) (see Usage report about an object)	information on usage	---	AII and CAMART