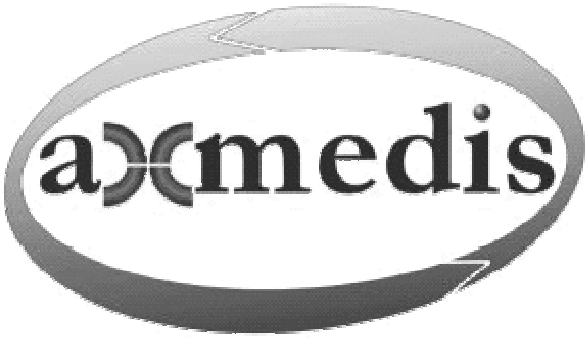


Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE2.1.1b
User Requirements and *use cases*

Version: 1.6-Public
Date: 14/01/2005
Responsible: DSI

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

Deliverable Number: DE2.1.1
Contractual Date of Delivery: see annex I
Actual Date of Delivery: 05-01-2005
Work-Package contributing to the Deliverable: WP2
Task contributing to the Deliverable: all of WP2
Nature of the Deliverable: document
Author(s): all

Abstract:
This document reports the requirements 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.

Table of Content

1 EXECUTIVE SUMMARY AND REPORT SCOPE 9

2 STRUCTURE OF USE CASES 11

2.1 STRUCTURE OF USE CASES 11
2.2 USE CASE AND SCENARIO DIAGRAM: SHAPES AND SEMANTICS 11

3 GENERAL USE CASES 13

3.1 MACRO-FUNCTIONALITY 13
3.1.1 Automatic collection of content into local AXMEDIS Database from proprietary CMS 13
3.1.2 Querying for AXMEDIS objects and Selection creation 14
3.1.3 Automatic load (and update) of AXMEDIS objects into local AXDB from AXEPTool 16
3.1.4 Automatic protection of AXMEDIS objects 18
3.1.5 Automatic composition of AXMEDIS objects 19
3.1.6 Automatic formatting of AXMEDIS objects 20
3.1.7 Automatic publication of AXMEDIS objects on AXEPTool 21
3.1.8 Automatic programme and publication of AXMEDIS objects on distribution channels 22
3.1.9 Acquisition of AXMEDIS objects from the distributor 23
3.1.10 Viewing/Using of AXMEDIS objects 24

4 AXMEDIS OBJECT EDITING 25

4.1 AXMEDIS EDITORS, AS AUTHORING TOOLS 25
4.1.1 Creation of a new AXMEDIS object 25
4.1.2 Load and save AXMEDIS objects 25
4.1.3 Navigating through AXMEDIS objects 26
4.1.4 Adding AXMEDIS elements to an existing AXMEDIS object 27
4.1.5 Extracting AXMEDIS elements 28
4.1.6 Removing an element from an AXMEDIS Object 28
4.1.7 Moving an element within the AXMEDIS Object 29
4.1.8 Adding a resource 29
4.1.9 Managing/Modifying a resources 30
4.1.10 Navigating and understanding DRM rules and PAR 31
4.2 AXMEDIS INTERNAL VIEWERS 32
4.2.1 Invoking an internal viewer/editor 32
4.2.2 Managing a digital resource by respecting the DRM in an Internal Viewer/Editor 32
4.2.3 Closing an Internal viewer/editor 33
4.3 AXMEDIS TOOLS FOR USING/PRODUCING AXMEDIS OBJECTS IN OTHER CONTENT TOOLS 33
4.3.1 Invoking an external tool with a digital resource belonging to the AXMEDIS object 33
4.3.2 Managing the digital resource by respecting the DRM in an external tool 34
4.3.3 Closing an External Tool 35
4.3.4 Updating a digital resource modified by an external tool 36
4.3.5 Transferring a digital resource to an external tool 36

5 AXMEDIS PRODUCTION TOOLS 37

5.1 COMPOSITIONAL TOOLS 37
5.1.1 Compositional Engine 37
5.1.1.1 Automatic composition 37
5.1.1.2 Compositional Engine verifies the compatibility of DRM associated with digital resources 37
5.1.1.3 Compositional Engine verifies the rights of digital resources 38
5.1.1.4 Compositional Engine embeds a digital resource in the new AXMEDIS object 38
5.1.1.5 Compositional Engine generates a new AXMEDIS object 39
5.1.1.6 Compositional Engine requires the Fingerprint estimation of a digital resource 39
5.1.1.7 Compositional Engine requires the Adaptation of a digital resource 40
5.1.1.8 Compositional Engine requires the Protection of the new AXMEDIS object 40
5.1.1.9 Compositional Engine merges component's DRM/PAR rules into a new AXMEDIS object 40
5.1.2 Composition Rules Editor 41
5.1.2.1 Create a new compositional rule 41
5.1.2.2 Search and Select a compositional rule 41

5.1.2.3 Activating a compositional rule 42
5.1.2.4 Removing a compositional rule 42
5.1.2.5 Debugging a compositional rule 43
5.2 FORMATTING TOOLS 43
5.2.1 Formatting Engine 43
5.2.1.1 Automatic formatting 43
5.2.1.2 Formatting Engine verifies the compatibility of DRM associated with digital resources 44
5.2.1.3 Formatting Engine verifies the rights of digital resources 44
5.2.1.4 Formatting Engine embeds a formatted digital resource in a new AXMEDIS object 45
5.2.1.5 Formatting Engine generates a new AXMEDIS object 45
5.2.1.6 Formatting Engine requires the Fingerprint estimation of a digital resource 45
5.2.1.7 Formatting Engine requires the Adaptation of a digital resource 46
5.2.1.8 Formatting Engine requires the Protection of the new formatted AXMEDIS object 46
5.2.1.9 Formatting Engine calls an External Tool to execute formatting operations 47
5.2.1.10 Formatting Engine merges DRM/PAR rules 47
5.2.2 Formatting Rules Editor 48
5.2.2.1 Create a new formatting rule 48
5.2.2.2 Search a rule 48
5.2.2.3 Activating a formatting rule 48
5.2.2.4 Removing a formatting rule 49
5.2.2.5 Debugging a formatting rule 49

6 AXMEDIS WORKFLOW 50

6.1 WORKFLOW SCENARIOS 50
6.2 CONTROLLING AND SUPERVISING LOCAL AXMEDIS TOOLS 60
6.2.1 General WorkFlow Use Cases 60
6.2.1.1 Create NPD Workspace 60
6.2.1.2 Add components to the NPD 60
6.2.1.3 Edit information of the NPD 60
6.2.1.4 Delete information of a NPD 61
6.2.1.5 Show information regarding components of a NPD 61
6.2.1.6 Delete a NPD 62
6.2.1.7 Search a NPD 62
6.2.1.8 Track Component 62
6.2.1.9 Identify the CPA for a NPD 63
6.2.1.10 Timestamp Generator 63
6.2.1.11 Generate Versions 63
6.2.1.12 List of Work 64
6.2.1.13 Select a Work Item from the List of Work 64
6.2.1.14 Complete a task of a work Item 65
6.2.1.15 Distribute the assigned Work to process and people 65
6.2.1.16 Change State/Phase of a Task for a work Item 66
6.2.1.17 Notification of information to a personnel for a task of a work 66
6.2.1.18 Global Viewer of all information of a NPD 66
6.2.1.19 Check-in task performed by manual operator 67
6.2.1.20 Check-out task performed by manual operator 67
6.3 CONTROLLING AND SUPERVISING AXMEDIS OBJECT LIFE IN AXMEDIS COMPLIANT FACTORIES 67

7 AXMEDIS OBJECT ACQUISITION FROM CMS 70

7.1 AUTOMATIC GATHERING OF CONTENT, COLLECTOR ENGINE 70
7.1.1 Setup for metadata mapping 70
7.1.2 Setup for content crawling 70
7.1.3 Define what content to acquire from Crawled Integrated Database 71
7.1.4 Start content crawling 71
7.2 FINGERPRINT EXTRACTOR AS A COLLECTION OF COLLECTOR ENGINE PLUG-INS FOR EXTRACTING FEATURES 72
7.2.1 Calculating content descriptors/fingerprint during crawling 72

8 AXMEDIS DATABASE 74

8.1 MANAGING A DATABASE OF AXMEDIS OBJECTS 74
8.1.1 Administer Objects in the AXMEDIS DB 74
8.1.2 Administer User in the AXMEDIS DB 74
8.1.3 Accessing a specific version of an AXMEDIS object 74
8.1.4 Removing last version of an AXMEDIS object 75
8.1.5 Removing an AXMEDIS object 75

8.1.6 User Management 76
8.1.7 User Groups Management 76
8.2 MAKING QUERIES INSIDE DATABASES OF AXMEDIS OBJECTS AND INSIDE THE OBJECTS 77
8.2.1 Querying for AXMEDIS objects and inside objects 77
8.2.2 Querying for AXMEDIS from Clients 78
8.2.3 Bookmark a query 79
8.2.4 Retrieve a bookmarked query 79
8.2.5 Organize bookmarked queries 80
8.2.6 Save an incomplete query 80
8.2.7 Retrieve an incomplete query 81

9 AXMEDIS AXEPTOOLS FOR P2P DISTRIBUTION ON B2B 82

9.1 AXEPTOOL FOR P2P ON B2B 82
9.1.1 Discovery and connection of peers on B2B P2P network 82
9.1.2 Report P2P downloads/uploads network traffic 82
9.2 PUBLICATION AND LOADING AXMEDIS OBJECTS OF AXEPTOOL 83
9.2.1 Creation of a publishing rule for the AXEPTool 83
9.2.2 Automatic publication of a selection of objects on the AXEPTool 83
9.2.3 Automatic updating of a modified object on the AXEPTool 84
9.2.4 Automatic publication of a not protected object on the AXEPTool 86
9.2.5 Manual Publication of AXMEDIS Objects with the AXEPTool 87
9.2.6 Producing a query to search on the AXEPTool 87
9.2.7 View/Manage query results coming from the AXEPTool 87
9.2.8 Active query pool management for the AXEPTool 88
9.2.9 Downloading an AXMEDIS object 88
9.2.10 Automatic downloading of a selection of objects available in the P2P network 89
9.2.11 Refining the selection (Active Selections) for the AXEPTool 90
9.2.12 Automatic loading new versions of AXMEDIS Objects for the AXEPTool 91
9.2.13 Automatic loading new AXMEDIS Objects with the AXEPTool 92
9.2.14 Manual Loading of AXMEDIS Objects with the AXEPTool 92
9.2.15 Creation of a loading rule for the AXEPTool 93
9.2.16 Preview an AXMEDIS object content coming from AXEPTool 93
9.2.17 Feedback toward the workflow system 94

10 PROGRAMME AND PUBLICATION ENGINE TOOLS 95

10.1 PROGRAMME AND PUBLICATION RULES PRODUCTION 95
10.2 PROGRAMME AND PUBLICATION RULES EDITING 96
10.3 ACTIVATION OF PROGRAMME AND PUBLICATION RULES 97
10.4 LAUNCH OF PROGRAMME AND PUBLICATION RULES FROM WORKFLOW 97
10.5 TRIAL PRE-ACTIVATION OF PROGRAMME AND PUBLICATION RULES 99
10.6 LAUNCH OF TRIAL PROGRAMME AND PUBLICATION RULES FROM WORKFLOW 99

11 AXMEDIS AXEPTOOLS FOR SATELLITE DATA BROADCAST ON B2B 101

11.1 AXMEDIS B2B CLIENT APPLICATION 101
11.1.1 B2B Client Installation 101
11.1.2 B2B Client Customization 101
11.1.3 B2B Client Registration 102
11.2 ENABLING A B2B RECEIVING STATION 102
11.3 DOWNLOADING AXMEDIS OBJECTS FROM AXEPTOOL BY USING SATELLITE DATA BROADCAST ON B2B 103
11.3.1 Pushing an AXMEDIS Object by B2B Carousel 103
11.3.2 Updating AXMEDIS Content by B2B Carousel 104
11.4 AUTOMATIC CONTENT RECEPTION VIA SATELLITE 104
11.5 CONTENT DELIVERY VIA SATELLITE 105
11.6 CONTENT PROTECTION FOR SATELLITE DISTRIBUTION 106

12 AXMEDIS PROTECTION TOOLS 107

12.1 SUPER AXCS 107
12.1.1 AXMEDIS Registration of AXCSs 107
12.1.2 Tool/device off-line registration 107
12.1.3 AXMEDIS Object ID Generation 107

12.1.3.1 Generation of unique Object ID 108
12.1.4 Global Object List WEB Service 108
12.1.4.1 Search of AXMEDIS Objects 108
12.1.5 Super AXCS Collector 109
12.1.5.1 On-line transfer between AXCS and Super AXCS 109
12.1.5.2 Off-line synchronization between AXCS and Super AXCS 109
12.2 AXMEDIS CERTIFIER AND SUPERVISOR 110
12.2.1 AXMEDIS Registration Service 110
12.2.1.1 End User registration in a distribution channel 110
12.2.1.2 End User registration in a different distribution channel 111
12.2.1.3 Registration of a new Teacher/School or Student 112
12.2.1.4 Registration of an old User of the Channel on AXMEDIS 113
12.2.1.5 User password modification 113
12.2.2 AXMEDIS Certification and Verification 114
12.2.2.1 Authentication of a Device 114
12.2.2.2 Certification of AXMEDIS Tool and User 115
12.2.2.3 Verification of AXMEDIS users using AXMEDIS tools 117
12.2.2.4 Verification of AXMEDIS users using AXMEDIS tools on a Device during content consumption inside a domain 118
12.2.3 AXMEDIS Supervisor 119
12.2.3.1 User blocking 119
12.2.3.2 User unblocking 120
12.2.3.3 Tool blocking 120
12.2.3.4 Tool unblocking 121
12.2.3.5 AXMEDIS Protection Information delivery 121
12.2.3.6 Storage of protection information of an AXMEDIS Object to the AXCS 122
12.2.3.7 Requesting of protection information of an AXMEDIS Object 122
12.2.4 AXMEDIS Reporting Web Service 122
12.2.4.1 Object usage reporting 122
12.2.5 Accounting Manager and Reporting Tool 123
12.2.5.1 List of all operations performed on an object 123
12.2.5.2 List of all operations performed by a user 123
12.2.5.3 Usage report about an object 123
12.2.5.4 Usage report about a distributor 124
12.2.5.5 Usage report about a provider 124
12.2.5.6 List objects for which an administrative account can be requested 124
12.2.5.7 Listing AXMEDIS clients of a distributor/channel 125
12.2.5.8 Listing distributors 125
12.2.6 AXCS Synchronizer 126
12.3 PROTECTION TOOL ENGINE 126
12.3.1 Content protection 127
12.3.2 Create a new protection rule 128
12.3.3 Search and Select a protection rule 129
12.3.4 Activating a protection rule 129
12.3.5 Removing a protection rule 130
12.3.6 Debugging a protection rule 130
12.3.7 Editing protection rules 130
12.3.8 Printing protection rules 131
12.4 ADMINISTRATIVE INFORMATION INTEGRATOR 133
12.4.1 Integrating Distributor administrative information of the basis of End User actions 133
12.4.2 Integrating Collecting Society administrative information of the basis of End User actions 134
12.4.3 Distributor asks for administrative information 135
12.4.4 Administrative information retrieval for distributors 137
12.4.5 Administrative information retrieval for collecting societies 137
12.5 PROTECTION MANAGER SUPPORT/SERVER GENERAL 138
12.5.1 Protection Manager Support / Server 138
12.5.1.1 Consumption of a protected and governed AXMEDIS object in a connected environment 138
12.5.1.2 Consumption of a protected and governed AXMEDIS object in a unconnected environment 138
12.5.1.3 Protection of an AXMEDIS object 139
12.5.1.4 Protection and association of licenses of/to an AXMEDIS object 139
12.5.1.5 Renewal of IPMP information after the detection of a succeed attack (connected) 140
12.5.2 DRM Support 140
12.5.2.1 License creation for new content 140
12.5.2.2 License creation for cross-media content 141

12.5.2.3 License migration 141
12.5.2.4 User authorisation 142
12.5.2.5 Navigation of licensing information 142
12.5.2.6 Rights Expression Translator 142
12.6 ENCRYPTION/DECRYPTION SUPPORT 143
12.6.1 Encryption 143
12.6.2 Decryption 143
12.6.3 Encryption of symmetric key 143
12.6.4 Decryption of symmetric key 144
12.6.5 Storage of security information 144
12.6.6 Retrieval of security information 144

13 AXMEDIS PLAYER 145

13.1 AXMEDIS PLAYER ON PC, TABLET PC 145
13.1.1 Content Recording for Playtime Shift 145
13.1.2 Fast-forward of Content in Internal Players/Viewers 146
13.1.3 Local adaptation of Content in Internal Players/Viewers 146
13.1.4 Annotate for personal use 147
13.1.5 Local User Profiles 148
13.1.6 History of the last played contents 148

14 AXMEDIS FOR DISTRIBUTION VIA INTERNET 148

14.1 BACK OFFICE MANAGEMENT 148
14.1.1 Creating a New Mediaclub 148
14.1.2 Mediaclub Setup 149
14.1.3 Mediaclub Accounts and Permission Management 149
14.1.4 Mediaclub Project Uploading and publishing contents 150
14.1.5 Mediaclub Project Acquiring AXMEDIS content 150
14.1.6 Mediaclub Project define payment gateway entry 151
14.1.7 Mediaclub Shop payment Management 151
14.1.8 Mediaclub Shop Management refound a transaction 151
14.2 END USER CLIENT CONFIGURATION 152
14.2.1 User Software Installation 152
14.2.2 User Registration 152
14.3 USER LOGIN 153
14.3.1 Authentication trough AXMEDIS client 153
14.3.2 Authentication trough an external SSO system 153
14.4 CATALOGUE BROWSING 154
14.4.1 Catalogue Listing 154
14.4.2 Catalogue Searching 155
14.4.3 Available resources listing 155
14.4.4 Content Access 156
14.4.5 User Page 156
14.5 CATALOGUE CONTENT PURCHASE 157
14.5.1 Content Fetching 157
14.5.2 User Authentication Form 157
14.5.3 Catalogue Content Transaction 158
14.5.4 Content Access 159
14.5.5 Content Preview 160
14.5.6 License Acquisition 160
14.5.7 Multi-device license activation and back-up 161
14.5.8 Pre-ordering and registration for a group of students 161
14.6 BUSINESS MODELS 162
14.6.1 Rental 162
14.6.2 pay per download 162
14.6.3 Sell-through 163
14.6.4 subscription 163
14.6.5 pay per minute 164
14.6.6 pay per Kb downloaded 164
14.6.7 pay per day 164

14.6.8 pay per credits 165
14.6.9 Grouped licenses 165
14.6.10 Packaged offers 166
14.7 ADVANCED PAYMENT METHODS 166
14.7.1 Gift Certificates 166
14.7.2 Wallet 168

15 AXMEDIS FOR DISTRIBUTION TOWARDS MOBILES 169

15.1 GENERAL ASSUMPTIONS AND NOTES TO ARCHITECTURE 169
15.2 USE CASES 170
15.2.1 Transcoding New Content 170
15.2.2 The APS Loads the Content Tree 171
15.2.3 Subscriber Browses the Content Tree 172
15.2.4 The Subscriber Samples Content 173
15.2.5 The Subscriber Purchases Content 174

16 AXMEDIS FOR DISTRIBUTION TOWARDS I-TV 174

16.1 USER TERMINAL INSTALLATION AND CONFIGURATION 174
16.1.1 User Hardware Installation 174
16.1.2 User Software Installation 175
16.1.3 User Registration 176
16.1.3.1 Application Selection 176
16.1.3.2 User Profiling 177
16.2 CONTENT LISTING 177
16.2.1 Content Web Listing 177
16.2.2 Content Carousel Listing 178
16.3 CONTENT VOTING 179
16.4 CONTENT SELECTION 179
16.4.1 Manual Content Selection 179
16.4.2 Automatic Content Selection 179
16.5 CONTENT RECEPTION 180
16.6 CONTENT REPARATION 180
16.7 CONTENT ACCESS 181
16.8 CONTENT PREVIEW 181
16.9 LICENSE ACQUISITION 182
16.9.1 User Identification 183
16.9.2 Billing 183
16.10 CONTENT BACKUP 184
16.11 CONTENT RESTORE 184
16.11.1 Cache Preloading 185
16.12 CACHE CLEANING 185
16.13 CACHE-BASED PERSONALISED CONTENT DISTRIBUTION SPECIFIC USE CASES 186
16.13.1 Automatic Content Access Set Up 186
16.13.2 AXMEDIS Channel personalisation 186
16.13.3 Automatic Content Access 186
16.13.4 AXMEDIS Channel PVR functionalities 187

17 AXMEDIS FOR DISTRIBUTION TO PDA VIA KIOSKS 187

17.1 CONTENT CATALOGUE CREATION 187
17.2 CONTENT CATALOGUE LOADING (PUBLICATION) 188
17.3 CONTENT CATALOGUE LOADING UPDATE 189
17.4 KIOSK START-UP 190
17.5 USER REGISTRATION TO KIOSK 190
17.6 USER LOGIN 192
17.7 CONTENT BROWSING & PREVIEWING 193
17.8 CONTENT SELECTION AND CHART MANAGEMENT 193
17.9 CHECK OUT PROCEDURE INITIATION 194
17.10 PURCHASING / ACQUIRING / RENTING 194
17.11 REPOSITORY SELECTION 195
17.12 DESTINATION TARGET IDENTIFICATION (UNIQUE ID FOR TARGET – WIFI) 195

17.13 DELIVERY TEMPLATE SELECTION (DEPENDING ON DEVICE) 196
17.14 DELIVERY FORMAT SELECTION (DEPENDING ON CONTENT) 196
17.15 DEVICE COMPATIBILITY (ROLL BACK IN CASE OF FAILURE) 197
17.16 STORAGE AVAILABILITY (ROLL BACK IN CASE OF FAILURE) 197
17.17 BILLING 198
17.18 DATA DELIVERY 198
17.19 CHECK OUT PROCEDURE CLOSURE 198
17.20 SUCCESSFUL DELIVERY CHECK (RECOVERY IN CASE OF FAILURE) 199
17.21 CONTENT FRUITION AFTER DOWNLOAD ON PDA OR MOBILE 201
17.22 CLIENT BASED CONTENT LICENSE VERIFICATION (ACCESS DENY IN CASE OF FAILURE) 201
17.23 APPLICATION FRONTEND INSTALLATION ON END USER DEVICE 202
17.24 USER PROFILE CHANGE 203
17.25 INTERFACE LANGUAGE SELECTION 204
17.26 USER DEVICE CONFIGURATION 205
17.27 CONTENT UPDATE (VIA SATELLITE) 206

1 Executive Summary and Report Scope

See DE2.1.1a, which is Part A of this document.

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 2-3 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 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 WP presents early requirements analysis with related tasks and a successive WP2.4 for its updating during the whole project when additional, or revised, detailed requirements will appear. 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:

- DE2.1.1 -- User Requirements and use cases (M3) – this deliverable contains the description of the user requirements and the corresponding use cases in UML, coming from WP2.1 and WP2.2;
- DE2.2.1 – Test cases and content description (M4) – this deliverable contain the description of the test cases for research functionalities and AXMEDIS tool validation, coming from WP2.2;
- DE2.4.1 – Requirements update (M18);
- DE2.3.1 – User Group Set up and analysis (M4). The analysis will be done on the basis of the curricula and the needs of the AXMEDIS project, to verify that all the aspects and user profiles and roles will be covered by the user group;
- DE2.3.2 – User Group Maintenance (M13).

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

WP2.1 -- Early Requirements Analysis -- collection of user requirements by using the expert user groups. The focus will mainly be on: content workflow, content management, content production, content searching, content rights management (licensing, formalising usage rules), content formatting in the various contexts (PC, mobile, i-TV, kiosk, PDA), user profiling, content composition, fingerprint,

watermark, indexing, querying, transaction models, push and pull balancing, etc. In addition, a more detailed analysis of the functionalities that could be useful in the above contexts will be done: query on technical aspects, content composition, content formatting, distribution, content exchanging, certification, supervision, etc. The use cases have to be collected by considering the points of view of content designers, multimedia producers, TISCALI, OD2, ANSC, AFI, ILABS, XIM, SEJER. In addition, EUTELSAT, HP, DSI, DIPITA, CPR, CRS4, IRC, UNIVLEEDS, EPFL, COMVERSE, ACIT, etc., will also collect this information from their experts by using specific interview based on guidelines produced by the consortium. A part of this information will be collected by reviewing the results of several past projects. In the analysis of requirements also those of the AXMEDIS partners and potential customers and SMEs in the respect of the WWW pages for getting general AXMEDIS services will be considered.

WP2.2 -- Use cases and test cases description -- this WP is devoted to the organisation of the requirements in terms of use cases and the corresponding identification and description of test cases. 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. The Content for the test cases will be collected and/or produced in WP8. The description about how the test cases will be selected and about which content will be suitable for that goal is reported in WP8. The use cases will be structured according to the UML model, including: name, ID, description, context assumptions (equipment, paradigm, location), actors (skill, age, instrument, paradigm), steps, variation, non functional aspects, content, interaction protocol, issues, etc. The test cases will be structured according to structure of the AXMEDIS framework and tools that will be developed in these 18 months of work. The model will be UML including: name, ID, description, functionality to be tested, context, partners involved, validator skill, data set needed, steps, expected results, variations, issues, additional activities to be considered, metrics to be used, etc. In this subWP, the targeted quality of use of the tools that will be developed during the project will be also defined in terms of metrics for usability. To this end users including the general public will be modelled based on the definition of the user requirements. The usability metrics will be focussed on extracting relevant drivers in the real environment of the application. Use cases and test cases for describing the interaction with the AXMEDIS services provided by the AXMEDIS portal will be separately described.

WP2.3 -- Set up and management of a AXMEDIS User Group -- a user group of experts will be set up. The members of the user group will receive updated information about the project evolution and will constitute a source for testing and validating the produced results. The user group has to present experts representing the different users of AXMEDIS tools at business and consumer levels. These are content producers, content integrators, content designers, usability experts, content distributors, content aggregators, publishers, etc. A separate deliverable has been produced reporting all the activities regarding the User Group.

For the terms and the definitions reported in this document please refer to the Specification Document Part J.

This document is comprised of two parts:

A) on requirements
B) on use cases

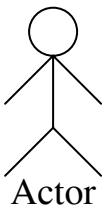
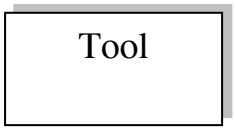
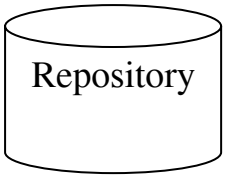
Test cases are reported in a different deliverable.

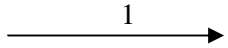
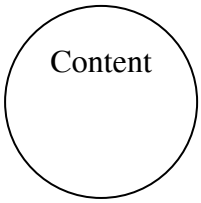
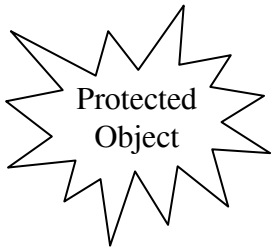
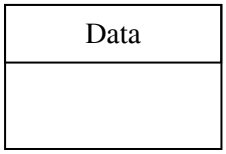
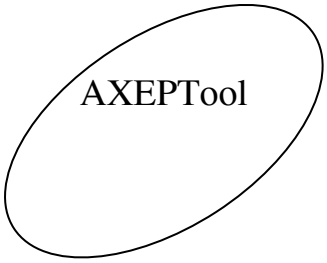
2 Structure of Use Cases

2.1 Structure of Use Cases

UCId	Unique identifier of the use case
Use case	Name of the use case
Description	Plain description of the use case activation, execution and termination
Actors	People, tools or entities involved in the use case, e.g. who (or what) activates the use case
Assumptions	Conditions which must be satisfied before use case activation
Steps	Step by step description of the use case activation, execution and termination
Post-conditions	Conditions which must be satisfied after use case termination
Variations	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
Asynchronous actions	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
Design suggestions	Useful hints or implications about the thought project structure regarding the use case
Issues	Possible issues, notes or annotations related to the use case implementation

2.2 Use Case and Scenario diagram: shapes and semantics

Shape	Name	Semantic
	Actor	The shape represents one of the Actor declared in the related use case or scenario
	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…
	Support	The shape represents the support whose name (defined in the specifications document) is contained within the shape, e.g. AXDBM, etc…
	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"

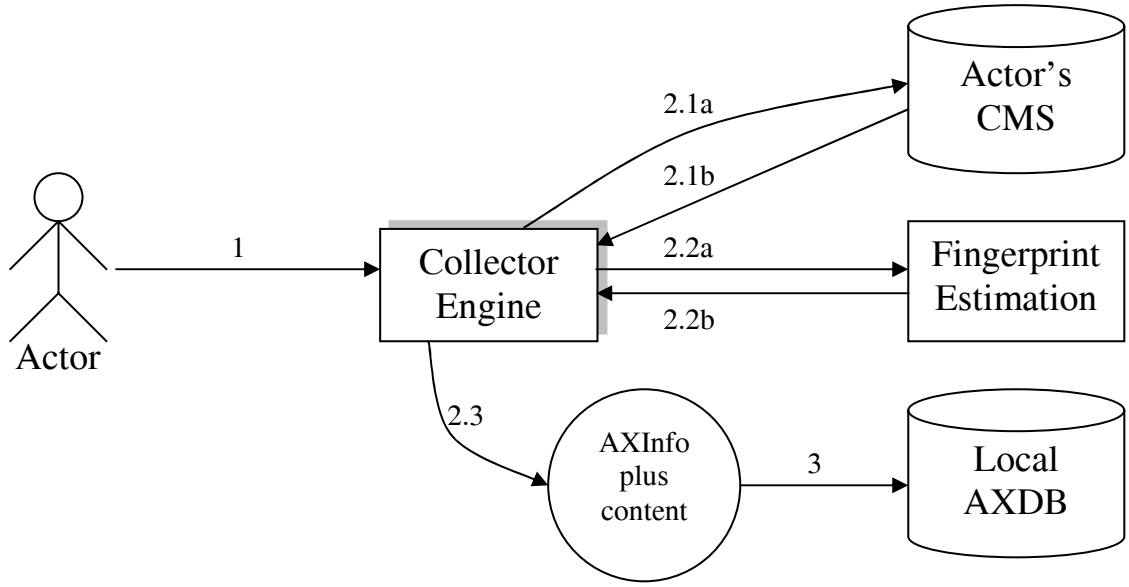
	Interaction	The shape represents an interaction between two modules, e.g. a call to an available function in another module, a data flow between tools, etc… The number above the arrow should correspond to a step of the related use case or scenario. If one step implies more than one interaction among actors, tools and supports than the label could be the number of the step plus a letter, e.g. 1a, 1b, etc… 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. 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…
	Unprotected AXMEDIS Object or Content	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
	Protected AXMEDIS Object	The shape represents a protected AXMEDIS object. The shape could contains the "name" of content with respect to the related use case
	Data	The shape represents data, other than content and protected/unprotected AXMEDIS object, used in the use case, e.g. selection, rule, etc…
	AXEPTool Distributed Database	The shape represents the AXEPTool in its meaning of distributed database as it is used in all other documents of AXMEDIS

3 General use cases

3.1 Macro-functionality

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.



UCId	UC3.1.1
Use case	Automatic collection of content into local AXMEDIS Database from proprietary CMS
Description	An Actor wants to import content from his/her CMS to his/her local AXMEDIS Database.
Actors	Creator, Producer
Assumptions	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.
Steps	1 The Actor, using Collector Engine, chooses which content shall be collected. The Actor could specify: o which content have to be collected o which fingerprints have to be estimated o mapping of metadata stored in the CMS 2 Collector Engine elaborates the request by: 2.1 picking up selected content from the Crawler Results Integrated Database 2.2 estimating specified fingerprints by using Fingerprint Estimation Tools as Plug-in for Collector Engine (through Collector Plug-in Manager) 2.3 putting together content, fingerprint and all other information needed to fulfil AXInfo schema within a simple AXMEDIS object 3 Collector Engine puts newly created AXMEDIS objects into the local AXDB
Post-conditions	None

Variations	• The Actor schedules the collection of content triggering it to a specified time and date • The Actor requests to collect content by using the AXMEDIS Workflow Manager • 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
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.2 Querying for AXMEDIS objects and Selection creation

Of course, 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):

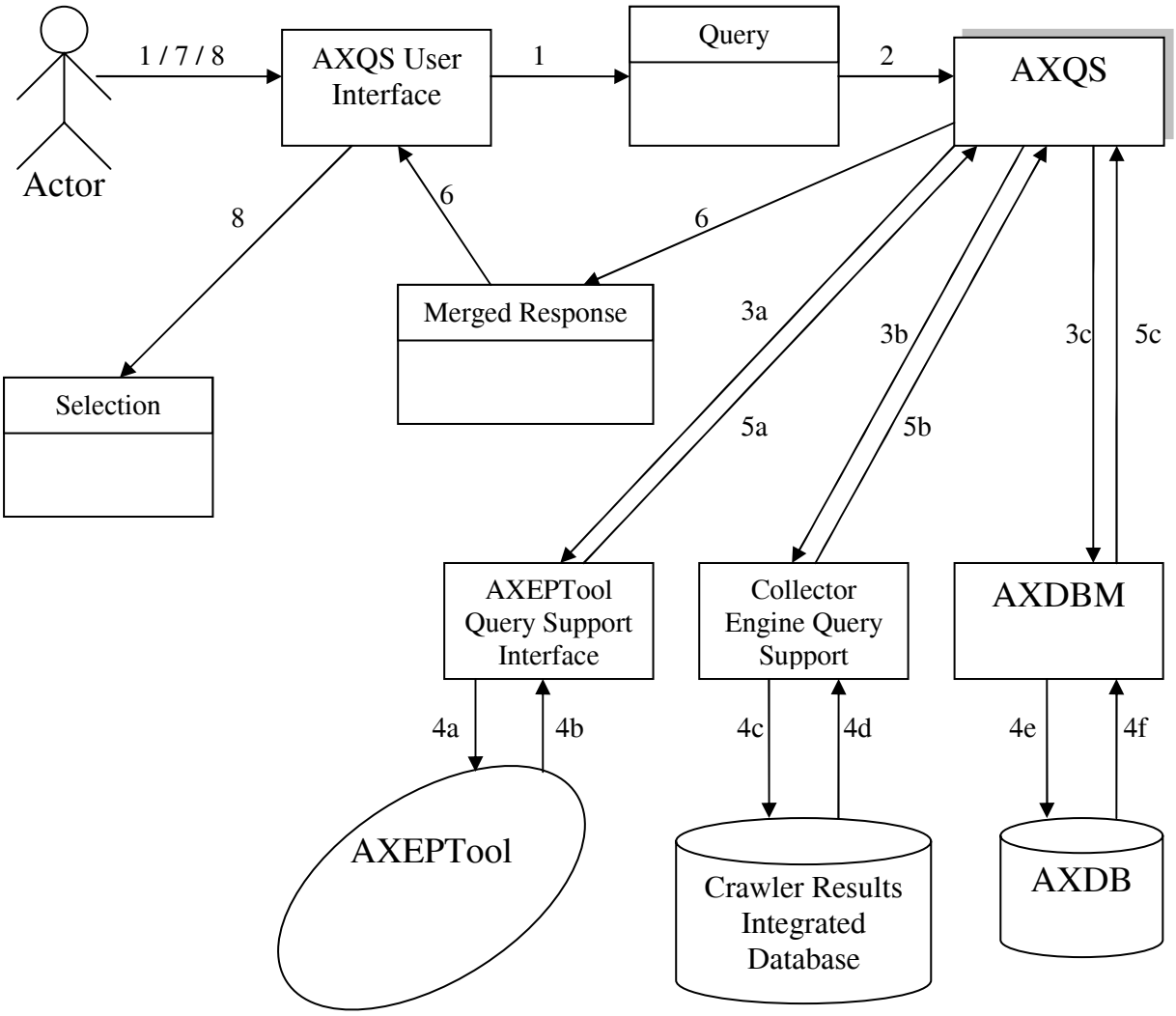
   S1={AXO1-1, AXO1-2, AXO1-3, AXO2-1, AXO2-2, AXO2-3, AXO2-4, AXO2-5, AXO2-6, AXO2-7, AXO2-8, AXO2-9, AXO3-1, AXO3-2, AXO4-1, AXO4-2, AXO4-3, AXO4-4, AXO4-5, AXO5-1}

   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. That is, if S2 is expanded approximately at the same time of its creation time the expanded Selection will probably equal to S1, conversely if S2 is expanded a long time after its creation the expanded Selection will probably be a different set of AXMEDIS objects.



UCId	UC3.1.2
Use case	Querying for AXMEDIS objects and Selection creation
Description	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
Actors	Aggregator, Publisher, Reseller, Retailer
Assumptions	None

Steps	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. 2 The Actor submits the queries previously composed 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 4 Each query interface (see step 3) looks for the required features in the corresponding domain 5 AXQS collects all the responses from the query interfaces 6 AXQS merges the results all together and return the complete list to the AXQS User Interface 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… 8 The Actor creates a new Selection (he/she could give it a name or a description) 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) 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)
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

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

UCId	UC3.1.3
Use case	Automatic load (and update) of AXMEDIS objects into local AXMEDIS Database from AXEPTool
Description	An Actor wants to load within his/her local AXMEDIS Database all AXMEDIS objects (published on AXEPTool) which belongs to a given Selection of AXMEDIS objects available on the AXEPTool network. Moreover, the Actor wants: • the system will update automatically the previously downloaded objects if they change on the AXEPTool network • the system will load automatically new objects which become element of the given Selection
Actors	Aggregator, Content Provider, Publisher, etc…
Assumptions	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 Publication/Loading Rules/Selections Editor (see use case "Querying for AXMEDIS objects and Selection creation")

Steps	1 Using the Publication/Loading Rules/Selections Editor, the Actor activates the previously created Selection submitting it to the P2P Active Selections. The Actor could also specify: o an activation period, i.e. a period of time for which the Selection will be elaborated 2 AXEPTool P2P Active Selection Engine elaborates the active Selections contained in the P2P Active Selections: 2.1 the AXEPTool P2P Active Selection Engine expands the Selection 2.2 for each object which belongs to the expanded Selection 2.2.1 AXEPTool P2P Active Selection Engine downloads the AXMEDIS object 2.2.2 the object is monitored by the AXEPTool Monitor 2.2.3 the object is stored on the AXEPTool IN AXMEDIS Database 3 After all objects have been downloaded, the Actor can try (e.g. play, run, visualize, etc…) each object accordingly to the related DRM rules 4 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 5 The Actor activates the new Selection submitting it to the AXEPTool Active Loading Rules/Selections still by using the Publication/Loading Rules/Selections Editor. The Actor could also specify o an activation period, maybe the same as step 1 6 The Loading Tool Engine of AXEPTool elaborates the active Selection: 6.1 the Loading Tool Engine of AXEPTool expands the Selection 6.2 for each object which belongs to the expanded Selection 6.2.1 load the AXMEDIS object into the local AXDB 7 Accordingly to the activation period given by the Actor, AXEPTool P2P Active Selection Engine and Loading Tool Engine of AXEPTool will elaborate the Selections 8 If a new version of one of the formerly downloaded object is published on AXEPTool network: 8.1 Publishing and Monitoring Objects is informed of this update by its identical counterpart which belongs to who has published the new version 8.2 Publishing and Monitoring Objects alerts AXEPTool P2P Active Selection Engine 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 8.4 if the object belongs 8.4.1 AXEPTool P2P Active Selection Engine will download again the object replacing the old version 8.4.2 AXEPTool IN AXDB will alert Loading Tool Engine of AXEPTool 8.4.3 Loading Tool Engine of AXEPTool will load the object to the local AXDB (supposing the related Selection still active) 9 If a new AXMEDIS object is published on AXEPTool network 9.1 Publishing and Monitoring Objects is informed by its identical counterpart which belongs to who has published the new object 9.2 Publishing and Monitoring Objects alerts AXEPTool P2P Active Selection Engine

Steps	9.3 AXEPTool P2P Active Selection Engine verifies if the newly published object belongs to one of the active Selections (verifying if it matches the features of a Query contained in an active Selection) stored in P2P Active Selections 9.4 AXEPTool P2P Active Selection Engine will download the new object 9.5 AXEPTool IN AXDB will alert Loading Tool Engine of AXEPTool 9.6 Loading Tool Engine of AXEPTool will load the object
Post-conditions	None
Variations	• Selections can be activated by using the AXMEDIS Workflow Manager • 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 • 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 and 9.
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.4 Automatic protection of AXMEDIS objects

UCId	UC3.1.4
Use case	Automatic protection of AXMEDIS objects
Description	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.
Actors	Aggregator, Author, Composer, Performer, Producer, Publisher and Reseller
Assumptions	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 Protection Tool User Interface and Rules Editor (see use case “Querying for AXMEDIS objects and Selection creation”)

Steps	1 The Actor composes protection rules by using Protection Tool Rules Editor. Such rules specifies: o the previously created Selection o PAR to be added o DRM rules to be added o how to protect those objects (encryption, specific algorithm, etc…) o the scheduling for the protection process 2 The Actor, through Protection Tool User Interface, submits the protection rules to the Protection Tool Engine Rules/Selections 3 AXPTE elaborates the Rules/Selections contained within the Protection Tool Engine Rules/Selections at the set time: 3.1 the AXPTE expands the Selection 3.2 for each object which belongs to the expanded Selection 3.2.1 AXPTE gets the AXMEDIS object by using the AXMEDIS Database Manager 3.2.2 AXPTE applies all requested protections to the object 3.2.3 AXPTE adds the required DRM rules and PAR through PMS and AXOM 3.2.4 AXPTE produces the license template related to PAR added to the object and send them to the License Generator using the PMS 3.2.5 AXPTE stores the protected AXMEDIS objects to the local AXDB using AXDBM
Post-conditions	None
Variations	• Protection rules can be activated by using the AXMEDIS Workflow Manager • The Actor could also request the immediate protection of a single AXMEDIS object using the Protection Tool Engine User Interface • 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 AXPTE when step 3 has been completed, e.g. via workflow functionalities
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.5 Automatic composition of AXMEDIS objects

UCId	UC3.1.5
Use case	Automatic composition of AXMEDIS objects
Description	An Actor wants to obtain, in an almost automatic way, a set of compounded AXMEDIS objects from a set of AXMEDIS objects
Actors	Aggregator
Assumptions	The Actor has previously created a set of Selections by using the AXQS User Interface integrated within the Compositional Rules Editor (see use case “Querying for AXMEDIS objects and Selection creation”).

Steps	1 The Actor, using Compositional Rules Editor, activates a compositional rule submitting the followings to the Active Compositional Rules: o a composition rule which could be created on-the-fly or chosen from the Repository of Compositional Rules o the previously created set of Selections which are the sets of AXMEDIS objects on which the rule will be applied o the right number and type of parameter (other than Selections) required by the compositional rule signature o scheduling information o etc… 2 The Compositional Engine elaborates the activated rules contained into the Active Compositional Rules according to the scheduling information. Such elaborations are made in respect of DRM rules of all involved objects. 3 Compositional Engine advises the Actor when the composition process is completed using workflow functionalities.
Post-conditions	All produced objects shall contain a new AXOID and information (contained in AXInfo) such as: their composition, usability, fingerprinting, etc…
Variations	• Compositional rules can be activated using the AXMEDIS Workflow Manager • The Actor could also request the composition of a set of AXMEDIS objects by using the Compositional Rules Editor, e.g. to test and verifies the behaviour of a rule • Instead of a rule, the Actor could gives to the Compositional Engine an object template which specifies how to glue together the AXMEDIS objects contained into the Selections
Asynchronous actions	None
Design suggestions	None
Issues	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 Compositional and Formatting use Cases.

3.1.6 Automatic formatting of AXMEDIS objects

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

Steps	1 The Actor, using Formatting Rules Editor, activates a formatting rule submitting the followings to the Active Formatting Rules: o a formatting rule which could be created on-the-fly or chosen from the Repository of Formatting Rules o the previously created Selection which is the set of AXMEDIS objects on which the rule will be applied o the right number and type of parameter (other than Selection) required by the formatting rule signature o scheduling information o etc… 2 The Formatting Engine elaborates the activated rules contained into the Active Formatting Rules according to the scheduling information. Such elaborations are made in respect of DRM rules of all involved objects. 3 Formatting Engine advises the Actor when the composition process is completed using workflow functionalities.
Post-conditions	All produced objects shall contain a new AXOID and information (contained in AXInfo) such as: their composition, usability, fingerprinting, etc…
Variations	• Formatting rules can be activated using the AXMEDIS Workflow Manager • The Actor could also request the formatting of an AXMEDIS objects by using the Compositional Rules Editor, e.g. to test and verifies the behaviour of a rule
Asynchronous actions	None
Design suggestions	None
Issues	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 Compositional and Formatting use Cases.

3.1.7 Automatic publication of AXMEDIS objects on AXEPTool

UCId	UC3.1.7
Use case	Automatic publication of AXMEDIS objects on AXEPTool
Description	An Actor wants to publish his/her AXMEDIS objects on the AXEPTool network to yield them available to the AXMEDIS community
Actors	Aggregator, Producer, etc…
Assumptions	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")

Steps	1 The Actor, using Publication/Loading Rules/Selections, activates the previously created Selection submitting it to the AXEPTool Active Publication Active Rules/Selections. The Actor could also specifies: o an activation period, which determines since when and for how long the objects of the Selection will be available on AXEPTool network o etc… 2 Publication Tool Engine of AXEPTool elaborates the active Selections contained in the AXEPTool Active Publication Active Rules/Selections according to activation periods: 2.1 the Publication Tool Engine of AXEPTool expands the Selection 2.2 for each object which belongs to the expanded Selection 2.2.1 Publication Tool Engine of AXEPTool publishes the object on the AXEPTool OUT AXDB 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 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 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. 6 Publication Tool Engine of AXEPTool controls if the object belongs to one of the active Selections. If it does: 6.1 Publication Tool Engine of AXEPTool update the object contained into the AXEPTool OUT AXDB with the new version of it 6.2 As described in Step 3 all the other peers in the network are advised 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" 6.4 The download will be monitored by the local AXEPTool Monitor 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. 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
Post-conditions	None
Variations	Publication of AXMEDIS objects on AXEPTool can be also made using the AXMEDIS Workflow Manager
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.8 Automatic programme and publication of AXMEDIS objects on distribution channels

UCId	UC3.1.8
Use case	Automatic programme and publication of AXMEDIS objects on distribution channels
Description	An Actor wants to create a programme to be published for a community of End-Users.

Actors	Publisher, Distributor
Assumptions	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
Steps	1 The Actor creates a Publication Rules specifying: o the Selection of AXMEDIS objects to be published o the publication periods, i.e. periods of time in which the AXMEDIS objects will be available for the End-User o optionally, a set of specific formatting rules (different from those usually used by the Programme and Publication Engine) 2 The Actor could save the Publication Rule into the Repository of Publication Rules/Selections 3 The Actor activates the Rule by submitting it to the Active Publication Rules/Selections related to a specific distribution channel (There are one Active Publication Rules/Selections and one Programme and Publication Engine for each distribution channel) 4 The Programme and Publication Engine related to the chosen distribution channel elaborates the active Rules: 4.1 Programme and Publication Engine expands the Selection contained in the Rule 4.2 before publication time, Programme and Publication Engine, using Formatting Engine, performs the needed adaptation on each object in the Selection. The adaptation request is made on the base of the default formatting rules of the Programme and Publication Engine, which are strictly connected to distribution channel characteristics through the Client View Profile associated to the Engine. If the Publication Rule contains some formatting rules those overrides the defaults 4.3 at publication time, Programme and Publication Engine uploads all the adapted objects on the Distribution Server 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)
Post-conditions	None
Variations	Publication process can be activated using the AXMEDIS Workflow Manager
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.9 Acquisition of AXMEDIS objects from the distributor

UCId	UC3.1.9
Use case	Acquisition of AXMEDIS objects from the distributor
Description	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")
Actors	End-User
Assumptions	The Actor is using an AXMEDIS complaint Client Viewer

Steps	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 2 The Actor submits the query 3 The Query Support for Client transmits the query to the Query Support for Distributions Channel and gets back the response 4 The Query Support for Client displays the response to the Actor. In particular it shows the following information: o name, e.g. song title, movie title o description, e.g. movie story/review o usability and related price, e.g. price for each kind of available action on the object o availability periods o relevant metadata o etc… 5 Before entirely downloading an object, the Actor could preview it if the object itself has been designed to allow this action 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") 7 The Actor downloads the objects on his/her terminal 8 The Actor pays the required fees and receives the licenses to use (accordingly to the acquired rights) the objects
Post-conditions	• The objects are stored on some storage devices directly reachable by the Actor • The related licenses are stored in the Protection Manager Support (server side) to which the Client Viewer refers • The Actor can use at any time and everywhere the object accordingly to the acquired rights
Variations	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
Asynchronous actions	None
Design suggestions	None
Issues	None

3.1.10 Viewing/Using of AXMEDIS objects

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

4 AXMEDIS Object Editing

4.1 AXMEDIS Editors, as authoring tools

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	1 The use case begins when an actor clicks on the "New object" buttons within the AXMEDIS Editor main window 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. 3 The system shows to the actor a hierarchical view of the object 4 The actor can add digital resources and can apply to them content processing algorithms such ass extractor of metadata, etc.
Post-conditions	None
Variations	• The actor could also click on "New…" within the "File" menu of the application • If the actor want to associate an AXMEDIS Object ID to the newly created object, the system has to request it to the AXMEDIS Object ID Generator through the Protection Manager Support
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	1 The Actor clicks on the "Open object" buttons within the AXMEDIS Editor main window 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 or in the Crawling Results Integrated Database. 3 The AXMEDIS Editor opens object using the AXOM 4 If errors did not occur at previous step, 4.1 The AXMEDIS Editor creates an Hierarchy Editor and Viewer showing the object 5 Else 5.1 The AXMEDIS Editor shows a dialog to inform the Actor about what did not work correctly 6 The Actor works with the object (see others use cases in this section) 7 The Actor clicks on the "Save object" buttons within the AXMEDIS Editor main window 8 The AXMEDIS Editor requests to the AXOM to save the object 9 The AXOM validates the software representation of the object 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 whatever was the source of the object (file-system or AXDB)
Post-conditions	None
Variations	• The actor could also click on "Open…" within the "File" menu of the application • 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 location. To do that step 7 should be replaced with: 7a. The Actor clicks on the "Save object as…" buttons within the AXMEDIS Editor main window 7b. The AXMEDIS Editor asks the Actor to input the new location which can be indifferently on the file-system or in the local AXDB Step 10 does not change except that instead of overwriting the old version the (modified) object is saved in the chosen location
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	• AXMEDIS Editor is opened • An object is opened within the AXMEDIS Editor

Steps	1 The use case begins when the Actor open an Hierarchy View of an opened object 2 The Hierarchy View shows a explorer-like tree view 3 The Actor could expand each tree node to view its children (if it contains someone) 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): 4.1 To manipulate the element/object structure (e.g. add a child/brother node, remove the element, etc…) 4.2 To open alternative element views (e.g. DRM Editor and View, Behaviour Editor and View, Metadata Editor and View, etc…) 4.3 Etc…
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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An Hierarchy View of the object is open
Steps	1 The use case begins when the Actor click with the right mouse button on an existing element 2 The Hierarchy View shows the proper context menu to the actor 3 The Actor chooses “Add element…” and then chooses the type of element he/she wants to add 4 If necessary, the Hierarchy View shows to the Actor a dialog to fill the element attributes and options 5 If the Actor confirms, the Hierarchy View requests to the AXOM to add the new element 6 If the Actor has the needed rights: 6.1 the AXOM adds the element to the AXMEDIS object 6.2 the Hierarchy View updates the displayed tree with the new added node 7 Else: 7.1 the AXOM informs the Hierarchy View about why the Actor can not add the element 7.2 Hierarchy View shows a dialog to the Actor with the received error
Post-conditions	None
Variations	• The Actor could click on “Add element…” within the “Edit” menu of the application instead of using the context menu • 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)
Asynchronous actions	None
Design suggestions	None

Issues	• Adds or insertion should be done without making validity controls because step-by-step consistency maintenance could bring usability lacks • By such way, a validity control is necessary when the actor wants to save the object

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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor
Steps	1 The use case begins when the Actor click with the right mouse button on an existing element 2 The Hierarchy View shows the proper context menu to the Actor 3 The Actor chooses "Extract element…" 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 5 If the Actor confirms: 5.1 the Hierarchy View requests to the AXOM to extract the element in the given location 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 6 The object will contains the selected element with all its relevant information, such as: DRM rules, metadata, etc…
Post-conditions	The new object must works in respect of all the DRM rules associated with the original element
Variations	• The Actor could click on "Extract element…"within the "Edit" (or "File"?) menu of the application instead of using the context menu • 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
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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An Hierarchy View of the object is open and the object contains at least one element (other than root element)

Steps	1 The use case begins when the Actor click with the right mouse button on an existing element 2 The Hierarchy View shows the proper context menu to the Actor 3 The Actor choose "Remove" 4 The Hierarchy View shows to the Actor a dialog to confirm the deletion 5 If the Actor confirms 5.1 the Hierarchy View requests to AXOM to remove the element 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 5.3 the informed views/editors updates the showed information
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	• Deletion should be done without making validity controls because step-by-step consistency maintenance could bring usability lacks • By such way, a validity control is necessary when the actor wants to save the object • The operations have to be allowed by the license if the AXMEDIS object is protected

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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An Hierarchy View of the object is open
Steps	1 The use case begins when the Actor clicks on an element and drags it 2 When the Actor drops the element, releasing the mouse button, the Hierarchy View controls if the chosen position is an allowed one. 3 If the position is a valid one: 3.1 the Hierarchy View request to the AXOM to move the element in the new position 3.2 the AXOM controls if the Actor is allowed to do the operation. In that case, the AXOM moves the element 4 Else, the Hierarchy View cancels the operation warning the Actor
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	1 The use case begins when the Actor chooses as type of element to be inserted "Resource" 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 3 The Hierarchy View requests to the AXOM to add the given resource 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 5 If the control succeeds: 5.1 If the Actor chooses to include the resource 5.1.1 The AXOM codifies the resource in base64 5.1.2 The AXOM includes the codified resource within the object 5.2 Else, the AXOM sets the "reference" attribute of the Resource element with value the location of the element
Post-conditions	
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	

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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An Hierarchy View of the object is open
Steps	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) 2 The Hierarchy View asks to the AXOM to look for an editor/viewer associated with resource mime type attribute 3 If there is no such editor/viewer, the Hierarchy View displays an error dialog 4 Else 4.1 The AXOM locks the Resource element 4.2 The AXOM certifies the editor/viewer 4.3 The AXOM runs the editor/viewer as an independent thread 4.4 The AXOM passes, through a secure channel, the extracted resource (view use case "Extract AXMEDIS elements") to the editor/viewer 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
Post-conditions	None

Variations	• The Actor could select a Resource element and than clicks on "Open resource…" within the "Edit" menu of the application instead of double-clicking on the element • 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): o Internal AXMEDIS Resource Editors/Viewers o External Editor/Viewer AXMEDIS Plug-ins o ActiveX Editor/Viewer AXMEDIS Plug-ins
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	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An Hierarchy View of the object is open
Steps	1 The use case begins when the Actor clicks with the right mouse button on an element 2 The Hierarchy View shows the proper context menu to the Actor 3 If enabled, the Actor chooses "DRM View…" 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: o understand which rights he/she can exercise on the object o understand which rights he/she can gain o 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 o look for predefined rules set or template o etc…
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

4.2 AXMEDIS Internal Viewers

4.2.1 Invoking an internal viewer/editor

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

UCId	UC4.2.2
Use case	Managing a digital resource by respecting the DRM in an Internal Viewer
Description	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
Actors	End User, Content Integrator, Content Distributor, Content Consumer
Assumptions	• An internal viewer has been invoked by the system

Steps	1 The use case begins when the user wants to perform a command on the digital resource 2 The system verifies the DRM of the resource (i.e. if the actor has the right to perform such command) 3 If the user is authorised 3.1 The internal viewer/editor performs the command 4 Else 4.1 The internal viewer/editor notifies a command failure message. 5 The Use Case ends.
Post-conditions	• The viewer continues to work
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	The DRM verification is performed by the AXOM

4.2.3 Closing an Internal viewer/editor

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

4.3 AXMEDIS tools for using/producing AXMEDIS Objects in other Content tools

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

UCId	UC4.3.1
Use case	Invoking external tools with a digital resource belonging to the AXMEDIS object
Description	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
Actors	End User, Content Integrator, Content Distributor, Content Consumer

Assumptions	• AXMEDIS Editor is open • An object is opened within the AXMEDIS Editor • An hierarchical view of the object is open
Steps	1 The use case begins when an actor clicks with the right mouse button on an resource 2 The system shows the proper context menu to the actor 3 The actor chooses "Open with…" 4 The proper viewer/editor is associated with the resource on the basis of MIME type 5 The system sends an opening authorization request to the PMS (via AXOM) 6 If PMS does not provide the authorization 6.1 The system displays an authorization failure message on screen 6.2 The Use Case ends 7 The system performs the verification of the AXMEDIS Editor 8 If the verification is not valid 8.1 The system displays an verification failure message on screen 8.2 The Use Case ends 9 The system calls the external tool associated with the resource 10 The external tool is configured by the AXMEDIS plug-in according to the DRM rules associated with the digital resource 11 The system sends the digital resource to the plug-in (via The Protection Processor of AXOM) 12 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) 13 The external tool shows the digital resource 14 The Use Case ends
Post-conditions	None
Variations	• Double click on the resource for automatic call of the tool by means MIME mechanism • A table with the available tools is associated with the resource, the system shows such table and the user selects the proper tool • 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
Asynchronous actions	None
Design suggestions	None
Issues	End User usually can only render an AXMEDIS objects so only external "viewers" should be invoked

4.3.2 Managing the digital resource by respecting the DRM in an external tool

UCId	UC4.3.2
Use case	Managing the digital resource by respecting the DRM in an external tool
Description	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
Actors	End User, Content Integrator, Content Distributor, Content Consumer
Assumptions	• An external tool has been invoked by the system • The external tool uses its AXMEDIS plug-in • The digital resource has been loaded in the AXMEDIS Data Model inside the plug-in and the resource is ready to be used. • The communication with the AXMEDIS Editor is active via plug-in

Steps	1 The use case begins when the user wants to execute a command provided by the external tool 2 The external tool communicates the request of command execution to the AXMEDIS plug-in 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. 4 If the actor is authorised 4.1 The AXMEDIS plug-in authorises the External tool to perform the command 5 Else 5.1 The AXMEDIS plug-in does not authorise the external tool to execute the command and notifies a command failure message. 6 The Use Case ends
Post-conditions	The external tool is still working
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	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.

4.3.3 Closing an External Tool

UCId	UC4.3.3
Use case	Closing an external tool
Description	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
Actors	End User, Content Integrator, Content Distributor, Content Consumer
Assumptions	• An external tool has been invoked by the system • The external tool uses the AXMEDIS plug-in
Steps	1 The use case begins when the user wants to quit the external tool 2 The user clicks with left mouse button on the close button of the external tool menu 3 If the digital resource is changed 3.1 The tool displays a dialog asking for the modification acceptance. 3.2 If the actor does not discard the modification 3.2.1 See the "Updating a digital resource modified by an external tool" Use Case 3.3 The tool is closed 3.4 The Use Case ends 4 The tool is closed 5 The Use Case ends
Post-conditions	• The modified digital resource has been updated in the AXMEDIS object • The external tool has been closed.
Variations	The use could quit the tool by selecting "Quit" in the menu bar.
Asynchronous actions	None
Design suggestions	None

Issues	None

4.3.4 Updating a digital resource modified by an external tool

UCId	UC4.3.4
Use case	Updating a digital resource modified by an external tool
Description	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
Actors	AXMEDIS Plug-in, system
Assumptions	• An external tool has been invoked by the system • The external tool uses the AXMEDIS plug-in • The digital resource has been modified
Steps	1 The use case begins when an updating command is invoked (to update a modified digital resource command) 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
Post-conditions	• The digital resource has been updated successfully inside the AXMEDIS object
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

4.3.5 Transferring a digital resource to an external tool

UCId	UC4.3.5
Use case	Transferring a digital resource to an external tool
Description	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
Actors	AXMEDIS Plug-in, AXMEDIS Editor, System
Assumptions	• An external tool has been invoked by the system • The external tool uses the AXMEDIS plug-in
Steps	1 The use case begins when the user wants to execute a command provided by the external tool 2 The external tool communicates the request of command execution to the AXMEDIS plug-in 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. 4 If all is correct and acceptable a digital resource can be transferred to the external tool 5 The AXMEDIS editor sends the digital resource to the AXMEDIS plug-in inside the external tool 6 The plug-in converts and transfers the digital resource to the external tool data model
Post-conditions	• The digital resource has been transferred successfully to the external tool data model
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	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 memory or via a temporary file depending on the relationships between the AXMEDIS Editor and the AXMEDIS Plug. All content processing operations, such as resource conversion, or others, have to be allowed by the license if the object is protected.

5 AXMEDIS Production Tools

5.1 Compositional Tools

5.1.1 Compositional Engine

5.1.1.1 Automatic composition

UCId	UC5.1.1.1
Use case	Automatic composition
Description	An Actor wants to compose automatically some AXMEDIS objects.
Actors	Content owner, Content Integrator, Content Distributor, AXMEDIS Workflow Manager
Assumptions	An active compositional rule is ready to be executed
Steps	1 The Use Case begins when the Compositional Engine receives a composition request coming from the AXMEDIS Workflow Manager or the internal scheduler activates a rule from the Active Composition Rules. 2 The internal scheduler sends a Composition Rule execution request and the corresponding rule to the Rule Executor. 3 The Compositional Engine executes the submitted rules by: 3.1 recovering all the specified AXMEDIS objects from AXMEDIS Database 3.2 verifying the compatibility of DRM and licensing 3.3 compounding AXMEDIS objects as described into selected rule 3.4 interacting with Fingerprint, Adaptation and Protection tools 3.5 storing all new created AXMEDIS objects into AXMEDIS Database (AXMEDIS Objects repository) 4 The Composition Engine sends an End process notification to the AXMEDIS Workflow Manager. 5 The Use Case ends
Post-conditions	None
Variations	The Actor can create his personalized compositional rule by using Compositional Rules Editor (see Use Case UCXXX).
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.2 Compositional Engine verifies the compatibility of DRM associated with digital resources

UCId	UC5.1.1.2
Use case	Compositional Engine verifies the compatibility DRM associated with digital resources

Description	A compositional 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 Compositional Engine verifies that DRM rules are compatible with the DRM rules and/or conditions specified in the rule
Actors	Compositional Engine
Assumptions	The DRM rules of digital resources related to the Selection are available
Steps	1 The Use Case starts when the Compositional Engine has to verify if the set of DRM rules match the DRM target specified in compositional rule. 2 If DRM are not compatible with the DRM and/or conditions specified in the rule. 2.1 The Composition fails and a composition failure notification is sent to the AXMEDIS Workflow Manager. 2.2 The Use Case ends. 3 The Compositional Engine continues the rule execution 4 The Use Case ends
Post-conditions	The current composition is interrupted.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.3 Compositional Engine verifies the rights of digital resources

UCId	UC5.1.1.3
Use case	Compositional Engine verifies the rights of digital resources
Description	A compositional rule could include the verification request of rights related to all digital resources. In this case, the Compositional Engine verifies that rights are compatible with the rights target specified in the rule.
Actors	Compositional Engine
Assumptions	The DRM rules of digital resources related to the Selection are available
Steps	1 The Use Case starts when the Compositional Engine has to verify if the set of rights match the rights specified in compositional rule. 2 If rights are not compatible with the rights specified in the rule. 2.1 The Composition fails and a composition failure notification is sent to the AXMEDIS Workflow Manager. 2.2 The Use Case ends. 3 The Compositional Engine continues the rule execution 4 The Use Case ends
Post-conditions	The current composition is interrupted.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.4 Compositional Engine embeds a digital resource in the new AXMEDIS object

UCId	UC5.1.1.4
Use case	Compositional Engine embeds a digital resource in the new AXMEDIS object
Description	Compositional Engine embeds physically or by reference one o more digital resource in the new AXMEDIS object.
Actors	Compositional Engine
Assumptions	None

Steps	1 The Use Case starts when the Compositional Engine has to embed a digital resource in the new AXMEDIS object 2 If the embedding option is "physically" 2.1 The composition engine sends an embedding request and the resource to the AXOM 3 Else the composition engine sends an embedding request and the reference of resource to the AXOM 4 The resource is embedded
Post-conditions	The resource or the reference is embedded in the AXMEDIS object
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.5 Compositional Engine generates a new AXMEDIS object

UCId	UC5.1.1.5
Use case	Compositional Engine generates a new AXMEDIS object
Description	Compositional Engine create one o more new AXMEDIS objects and assign them a new Object ID
Actors	Compositional Engine
Assumptions	None
Steps	1 The Use Case starts when the Compositional Engine create a new AXMEDIS object following a compositional rule 2 The composition engine asks for a new Object ID to the AXMEDIS OID Generator. 3 The ID is applied to the new object.
Post-conditions	The Object is created and a new ID has been assigned
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.6 Compositional Engine requires the Fingerprint estimation of a digital resource

UCId	UC5.1.1.6
Use case	Compositional Engine requires the fingerprint estimation of a digital resource
Description	If a fingerprint request for a digital resource is specified in the composition rule the Compositional Engine interacts with the Fingerprint tool asking for fingerprint estimation (via AXOM). The Fingerprint tool will return the content descriptors related to the digital resource.
Actors	Compositional Engine
Assumptions	The digital resource is available physically during the composition process
Steps	1 The Use Case starts when a fingerprint estimation request for the digital resource is specified in the composition rule. 2 The digital resource is sent to the Fingerprint tool 3 The Fingerprint tool returns the content descriptors associated with the digital resource. 4 The content descriptors are inserted as metadata associated with the digital resource.
Post-conditions	None
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.7 Compositional Engine requires the Adaptation of a digital resource

UCId	UC5.1.1.7
Use case	Compositional Engine requires the Adaptation of a digital resource
Description	If an adaptation request for a digital resource is specified in the composition rule the Compositional Engine interacts with the Adaptation tool (via AXOM). The Adaptation tool will perform the adaptation specified in the composition rule for the a given digital resource.
Actors	Compositional Engine
Assumptions	The digital resource is available physically during the composition process
Steps	1 The Use Case starts when an adaptation request for the digital resource is specified in the composition rule 2 The digital resource is sent to the adaptation tool 3 The Adaptation tool returns the adapted resource
Post-conditions	The initial digital resource is adapted on the basis of adaptation request specified in the rule
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.8 Compositional Engine requires the Protection of the new AXMEDIS object

UCId	UC5.1.1.8
Use case	Compositional Engine requires the protection of the new AXMEDIS object
Description	If a protection request for the new AXMEDIS object is specified in the composition rule the Compositional Engine interacts with the Protection tool (via AXOM). The Protection tool will create an AXMEDIS protected object.
Actors	Compositional Engine
Assumptions	The digital resource is available physically during the composition process
Steps	1 The Use Case starts when the rule executor has to perform a fingerprint estimation. 2 The digital resource is sent to the Fingerprint tool 3 The Fingerprint tool returns the content descriptors associated with the digital resource.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.1.9 Compositional Engine merges component's DRM/PAR rules into a new AXMEDIS object

UCId	UC5.1.1.9
Use case	Compositional Engine merges component's DRM/PAR rules into a new AXMEDIS object

Description	Compositional Engine create a new AXMEDIS objects and merge component's DRM/PAR rules to create a new DRM/PAR rule
Actors	Compositional Engine
Assumptions	None
Steps	1 The Use Case starts when the compositional engine has to generate the DRM/PAR for the composite AXMEDIS object 2 Compositional Engine merge component's DRM/PAR rules into the new AXMEDIS Objects 3 The actor can modify the DRM/PAR rule of the new AXMEDIS object without modify the DRM/PAR rules of the components, but in according to them. The verification can be invoked by exploiting services of the PMS client in conjunction with the AXOM.
Post-conditions	None
Variations	The DRM/PAR of the whole new object depends on what could be the intersection of DRM/PAR rules related to each component
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.2 Composition Rules Editor

5.1.2.1 Create a new compositional rule

UCId	UC5.1.2.1
Use case	Create a new compositional rule
Description	An Actor wants to create a new compositional rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	None
Steps	1 The Actor creates a Selection of digital resources by making queries to the AXMEDIS Database 2 The Actor rules how these resources have to be compound 3 The Actor stores the created rule into Composition Rules Database
Post-conditions	None
Variations	1) The Actor defines a Selection by writing in the rule the scripting code (Composition 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 (Composition Rule Language) or in a Visual way.
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.2.2 Search and Select a compositional rule

UCId	UC5.1.2.2
Use case	Search and Select a compositional rule
Description	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.
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	None

Steps	1 The Actor search into the Repository of Compositional Rules a specific compositional rule 2 The rules are ordered in some manner and simple queries can be performed 3 If the Actor founds the rule can : 3.1 Use it to create a compounded AXMEDIS object 3.2 Modify it 3.2.1 Then the Actor store the new rule into the Repository by Compositional Rules Editor 3.2.2 Use the new rule to create a compounded AXMEDIS object 4 If the Actor doesn't found the rule can create a new one
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.2.3 Activating a compositional rule

UCId	UC5.1.2.3
Use case	Activating a compositional rule
Description	An Actor wants to activate a compositional rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	The Composition Rule Editor can access to the Active Rules Repository
Steps	1 The Actor search into the Repository of Compositional Rules a specific compositional rule 2 If the Actor doesn't found the rule 2.1 The Actor can create a new one 3 The Actor selects "Activate Rule" function 4 The rule is put into the Active Rules Repository
Post-conditions	The rule is added to the set of Active Rule
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.2.4 Removing a compositional rule

UCId	UC5.1.2.4
Use case	Removing a compositional rule
Description	An Actor wants to remove a compositional rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	The Composition Rule Editor can access to the Active Rules Repository
Steps	1 The Actor requests the list of Active Rules in the Active Rules Repository 2 The Actor selects the active rule to be disabled 3 The Actor selects "Remove Rule" function 4 The rule is Removed
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.1.2.5 Debugging a compositional rule

UCId	UC5.1.2.5
Use case	Debugging/Simulation a compositional rule
Description	An Actor wants to debug a compositional rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	A composition rule is available.
Steps	1 The Use Case starts when the Actor wants to debug a rule 2 The Rule Editor enters in the Debugging/Simulation Mode 3 During the debugging mode the Actor can: 3.1 Check the statements of rule step by step 3.2 Control the values of current variables 3.3 Exit from the debugging mode
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2 Formatting Tools

5.2.1 Formatting Engine

5.2.1.1 Automatic formatting

UCId	UC5.2.1.1
Use case	Automatic formatting
Description	An Actor wants to format automatically some AXMEDIS objects.
Actors	Content owner, Content Integrator, Content Distributor, AXMEDIS Workflow Manager
Assumptions	An active formatting rule is ready to be executed
Steps	1 The Use Case begins when the Formatting Engine receives a formatting request coming from the AXMEDIS Workflow Manager or the internal scheduler activates a rule from the Active Composition Rules. 2 The internal scheduler sends a Formatting Rule execution request and the corresponding rule to the Rule Executor. 3 The Formatting Engine executes the submitted rules by: 3.1 recovering all the specified AXMEDIS objects from AXMEDIS Database 3.2 verifying the compatibility of DRM and licensing 3.3 creating the formatted AXMEDIS objects as described into selected rule 3.4 interacting with Fingerprint, Adaptation and Protection tools 3.5 storing all new created AXMEDIS objects into AXMEDIS Database (AXMEDIS Objects repository) 4 The Formatting Engine sends an End process notification to the AXMEDIS Workflow Manager. 5 The Use Case ends
Post-conditions	An AXMEDIS formatted object is produced.
Variations	• The Actor can create his personalized formatting rule by using Formatting Rules Editor • The formatting process can be executed by an external tool if specified in the formatting rule • The adaptation of resources could be performed by using external functionalities provided by external tools

Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.2 Formatting Engine verifies the compatibility of DRM associated with digital resources

UCId	UC5.2.1.2
Use case	Formatting Engine verifies the compatibility DRM associated with digital resources
Description	A Formatting 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 Formatting Engine verifies that DRM rules are compatible with the DRM rules and/or conditions specified in the rule
Actors	Formatting Engine
Assumptions	The DRM rules of digital resources related to the Selection are available
Steps	1 The Use Case starts when the Formatting Engine has to verify if the set of DRM rules match the DRM target specified in Formatting rule. 2 If DRM are not compatible with the DRM and/or conditions specified in the rule. 2.1 The Formatting process fails and a formatting failure notification is sent to the AXMEDIS Workflow Manager. 2.2 The Use Case ends. 3 The Formatting Engine continues the rule execution 4 The Use Case ends
Post-conditions	The current formatting process is interrupted.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.3 Formatting Engine verifies the rights of digital resources

UCId	UC5.2.1.3
Use case	Formatting Engine verifies the rights of digital resources
Description	A formatting rule could include the verification request of rights related to all digital resources. In this case, the Formatting Engine verifies that rights are compatible with the rights target specified in the rule.
Actors	Formatting Engine
Assumptions	The DRM rules of digital resources related to the Selection are available
Steps	1 The Use Case starts when the Formatting Engine has to verify if the set of rights match the rights specified in formatting rule. 2 If rights are not compatible with the rights specified in the rule. 2.1 The Formatting fails and a formatting failure notification is sent to the AXMEDIS Workflow Manager. 2.2 The Use Case ends. 3 The Formatting Engine continues the rule execution 4 The Use Case ends
Post-conditions	The current formatting is interrupted.
Variations	None
Asynchronous actions	None
Design suggestions	None

Issues	None

5.2.1.4 Formatting Engine embeds a formatted digital resource in a new AXMEDIS object

UCId	UC5.2.1.4
Use case	Formatting Engine embeds a formatted digital resource in the new AXMEDIS object
Description	Formatting Engine embeds one o more digital resource in the new AXMEDIS object.
Actors	Formatting Engine
Assumptions	None
Steps	1 The Use Case starts when the Formatting Engine has to embed the new formatted digital resource in the new AXMEDIS object 2 The composition engine sends an embedding request and the resource to the AXOM 3 The resource is embedded
Post-conditions	The resource is embedded in the new AXMEDIS object
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.5 Formatting Engine generates a new AXMEDIS object

UCId	UC5.2.1.5
Use case	Formatting Engine generates a new AXMEDIS object
Description	Formatting Engine create one o more new AXMEDIS objects and assign them a new Object ID
Actors	Formatting Engine
Assumptions	None
Steps	1 The Use Case starts when the Formatting Engine create a new AXMEDIS object following a formatting rule 2 The formatting engine asks for a new Object ID to the AXMEDIS OID Generator. 3 The ID is applied to the new object.
Post-conditions	The Object is created and a new ID has been assigned
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.6 Formatting Engine requires the Fingerprint estimation of a digital resource

UCId	UC5.2.1.6
Use case	Formatting Engine requires the fingerprint estimation of a digital resource
Description	If a fingerprint request for a digital resource is specified in the formatting rule the Formatting Engine interacts with the Fingerprint tool asking for fingerprint estimation (via AXOM). The Fingerprint tool will return the content descriptors related to the digital resource.
Actors	Formatting Engine
Assumptions	The digital resource is available physically during the composition process

Steps	1 The Use Case starts when a fingerprint estimation request for the digital resource is specified in the formatting rule. 2 The digital resource is sent to the Fingerprint tool 3 The Fingerprint tool returns the content descriptors associated with the digital resource. 4 The content descriptors are inserted as metadata associated with the digital resource. 5 The Use Case ends
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.7 Formatting Engine requires the Adaptation of a digital resource

UCId	UC5.2.1.7
Use case	Compositional Engine requires the Adaptation of a digital resource
Description	If an adaptation request for a digital resource is specified in the composition rule the Formatting Engine interacts with the Adaptation tool (via AXOM). The Adaptation tool will perform the adaptation specified in the formatting rule for the a given digital resource. The adaptation is based on a set of parameters such as: user profile, channel distribution and device profile, file format, etc…
Actors	Formatting Engine
Assumptions	The digital resource is available physically during the composition process
Steps	1 The Use Case starts when an adaptation request for the digital resource is specified in the formatting rule 2 The digital resource and the set of formatting parameters are sent to the adaptation tool 3 The Adaptation tool returns the adapted resource 4 The Use Case ends
Post-conditions	The initial digital resource is adapted on the basis of adaptation request specified in the rule
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.8 Formatting Engine requires the Protection of the new formatted AXMEDIS object

UCId	UC5.2.1.8
Use case	Formatting Engine requires the protection of the new formatted AXMEDIS object
Description	If a protection request for the new AXMEDIS object is specified in the composition rule the Compositional Engine interacts with the Protection tool (via AXOM). The Protection tool will create an AXMEDIS protected object.
Actors	Formatting Engine and Fingerprint Estimation Tools
Assumptions	The formatted digital resource is available physically during the composition process
Steps	1 The Use Case starts when the rule executor has to perform the protection of the formatted object 2 The formatted object is sent to the Protection tool 3 The Protection tool returns the protected AXMEDIS object 4 The Use Case ends

Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.1.9 Formatting Engine calls an External Tool to execute formatting operations

UCId	UC5.2.1.9
Use case	Formatting Engine calls an External Tool to execute formatting operations
Description	If a request of services provided by external tools is specified in the formatting rule the Formatting 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.
Actors	Formatting Engine
Assumptions	The digital resources to be formatted are available physically during the formatting process
Steps	1 The Use Case starts when the rule executor has to perform an external call to a formatting tool specified in the rule 2 The formatting engine sends the digital resources and parameters specified in the external call to the external tool 3 The external tool performs functions specified in the rule 4 The external tool returns formatted digital resources 5 The Use Case ends
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	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…
Issues	None

5.2.1.10 Formatting Engine merges DRM/PAR rules

UCId	UC5.2.1.10
Use case	Formatting Engine merges DRM/PAR rules
Description	Formatting Engine merges DRM/PAR rules of the digital resources into a new formatted AXMEDIS object
Actors	Formatting Engine
Assumptions	None
Steps	1 Formatting Engine create a new AXMEDIS objects following a formatting rule 2 Formatting Engine merge component's DRM rules (in terms of PAR or included licences, if any) into the new AXMEDIS Objects 3 The actor can modify the DRM/PAR rule of the new AXMEDIS object without modify the DRM/PAR rules of the components, but in according to them. The verification can be invoked by exploiting services of the PMS client in conjunction with the AXOM. 4 Algorithms of PAR adaptation will be provided to perform such operations. 5 The Use Case ends
Post-conditions	None
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.2 Formatting Rules Editor

5.2.2.1 Create a new formatting rule

UCId	UC5.2.2.1
Use case	Create a new formatting rule
Description	An Actor wants to create a new formatting rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	None
Steps	1 The Actor creates a Selection of digital resources by making queries to the AXMEDIS Database 2 The Actor rules how these resources have to be formatted 3 The Actor stores the created rule into Formatting Rules Database
Post-conditions	None
Variations	1) The Actor defines a Selection by writing in the rule the scripting code (Formatting 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 (Formatting Rule Language) or in a Visual way.
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.2.2 Search a rule

UCId	UC5.2.2.2
Use case	Search a rule
Description	An Actor wants to search a specific formatting rule
Actors	Content Integrator, Content Distributor
Assumptions	None
Steps	1 The Actor search into the Repository of Formatting Rules a specific formatting rule 2 If the Actor founds the rule can : 2.1 Use it to format an AXMEDIS object 2.2 Modify it 2.2.1 Then the Actor store the new rule into the Repository by Formatting Rules Editor 2.2.2 Use the new rule to format an AXMEDIS object 3 If the Actor doesn't found the rule can create a new one
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.2.3 Activating a formatting rule

UCId	UC5.2.2.3
Use case	Activating a formatting rule
Description	An Actor wants to activate a formatting rule

Actors	Content owner, Content Integrator, Content Distributor
Assumptions	The Formatting Rule Editor can access to the Active Rules Repository
Steps	1 The Actor search into the Repository of Formatting Rules a specific formatting rule 2 If the Actor doesn't found the rule 2.1 The Actor can create a new one 3 The Actor selects "Activate Rule" function 4 The rule is put into the Active Rules Repository
Post-conditions	The rule is added to the set of Active Rule
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.2.4 Removing a formatting rule

UCId	UC5.2.2.4
Use case	Removing a formatting rule
Description	An Actor wants to remove a formatting rule
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	The Formatting Rule Editor can access to the Active Rules Repository
Steps	1 The Actor requests the list of Active Rules in the Active Rules Repository 2 The Actor selects the active rule to be disabled 3 The Actor selects "Remove Rule" function 4 The rule is removed from the Active Rules Repository
Post-conditions	The rule is disabled
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

5.2.2.5 Debugging a formatting rule

UCId	UC5.2.2.5
Use case	Debugging/Simulating a formatting rule
Description	An Actor wants to debug a formatting rule to check the correctness and feasibility
Actors	Content owner, Content Integrator, Content Distributor
Assumptions	A formatting rule is available
Steps	1 The Use Case starts when the Actor wants to debug a rule 2 The Rule Editor enters in the Debugging Mode 3 During the debugging mode the Actor can: 3.1 Check the statements of rule step by step 3.2 Control the values of current variables 3.3 Exit from the debugging mode
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6 AXMEDIS Workflow

6.1 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.

He invokes the "create new NPD workspace" function by clicking on this button to define the product and the NPD for its development. A pop-up dialogue box appears to allow him to enter the basic details of the NPD (e.g. name, type, etc) and to select pre-defined templates.

The workflow manger (AXWFM) communicates with the AXMEDIS Object Manager (AXOM) through the AXMEDIS editor workflow plug-in, to generate a Process ID which is to be assigned to the new NPD.

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.

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.

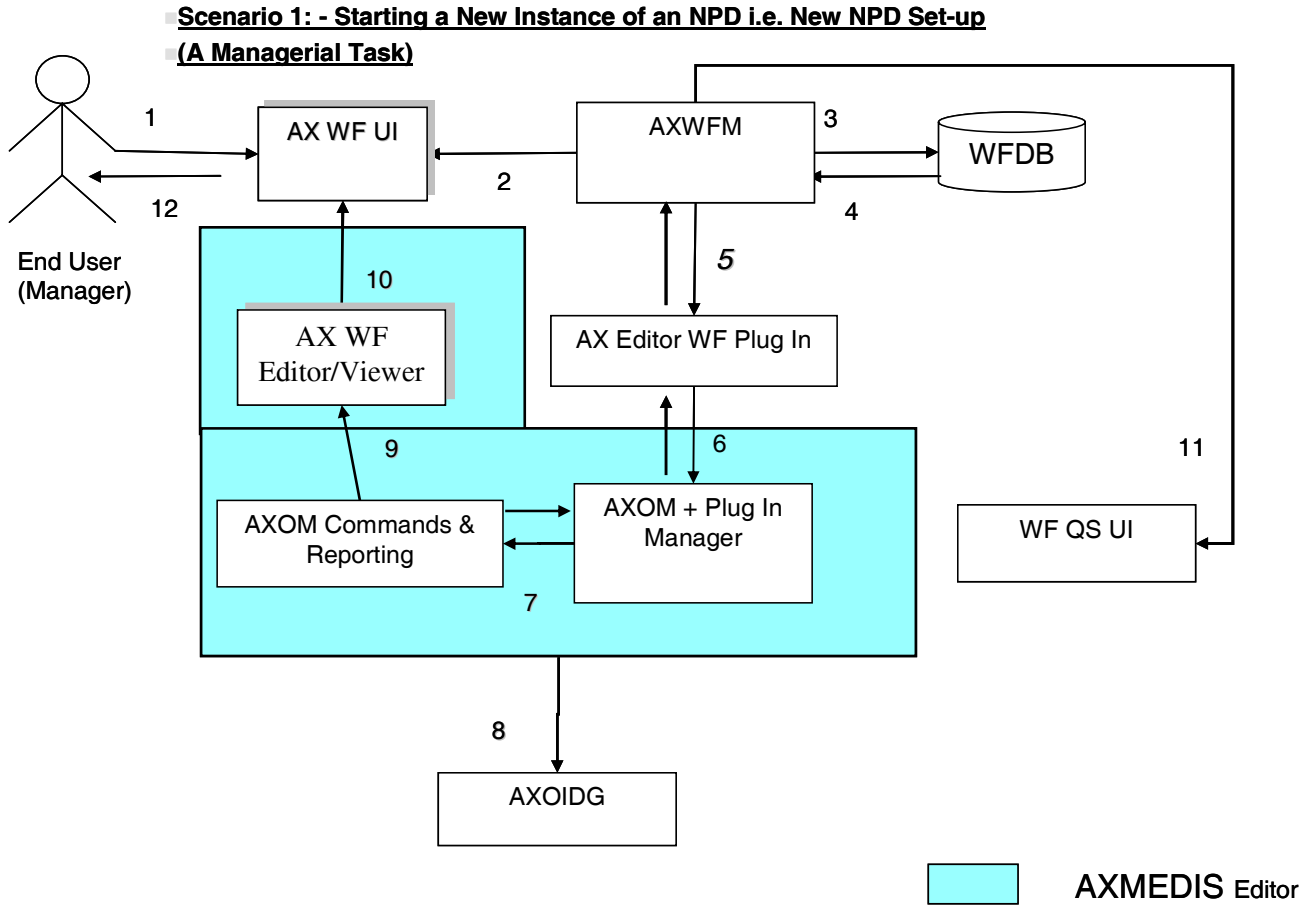
For example the user can add a task to the current NPD workspace and may designate its type as a "Formatting Task". The user then can add a person to the current workspace and assign his role to be, say, the "Technical Editor" responsible for the formatting Task. He can then link this person to the "Formatting

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. XPDL); 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.



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 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 and billing purposes has been provided.

The user invokes the list-work function by clicking on the button list-work supplying a workflow-instance_ID which supersedes which effectively represents a given NPD, 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).

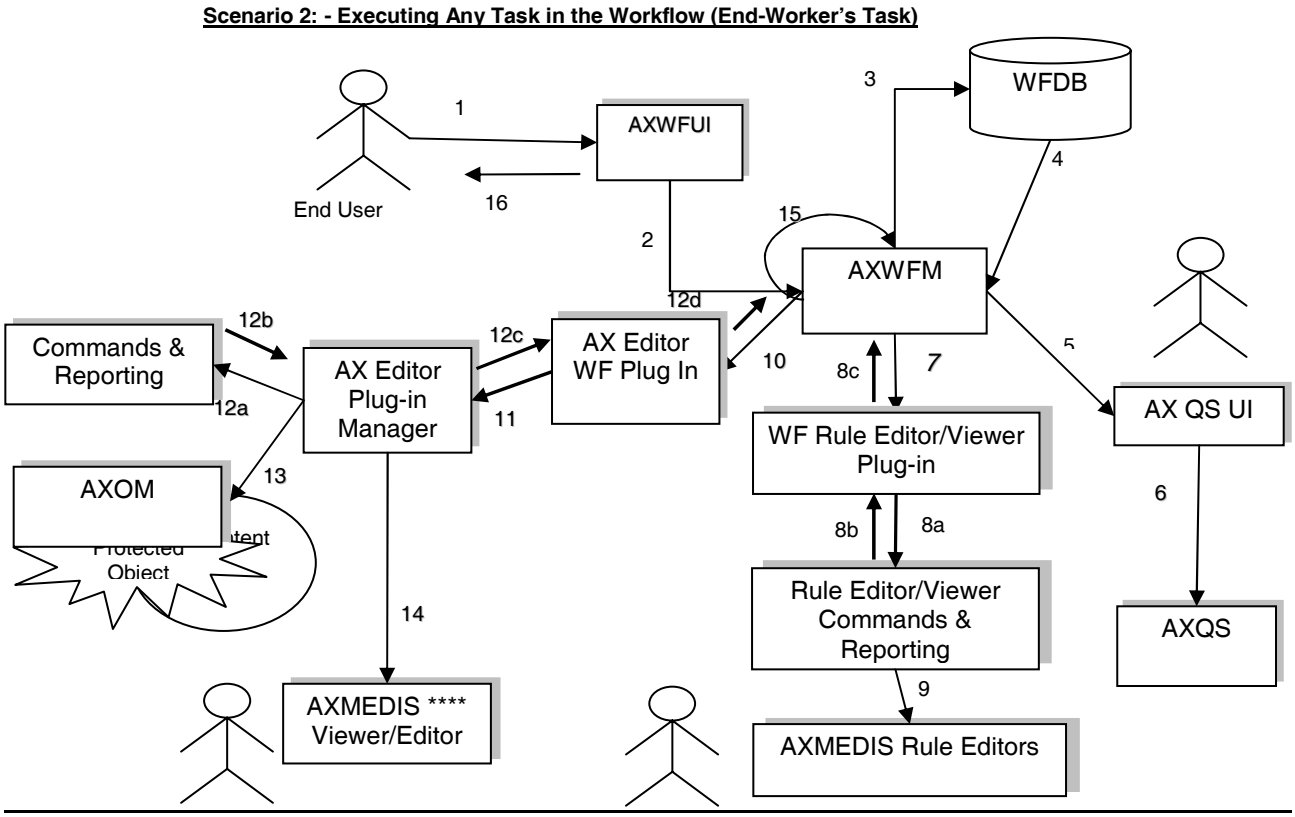
Based on the selected Task the workflow system launches the required tool using the appropriate Interface (e.g. Web-services) or plug-ins associated with that tool. If the tool is in the exclusive access area of the user, the "Check-in" and "Check-out" interfaces will be invoked.

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.



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: Invoking the AXEPTool (Publish)

This scenario describes the interaction between the workflow and the AXEPTool to share any AXMEDIS Object over the P2P network. There are two possible interaction scenarios between the workflow and the AXEPTool. On the one side, the interaction can be for uploading (Publishing) of some AXMEDIS Object(s), while on the other side the interaction can be for downloading (Loading) of an AXMEDIS Object. The Loading operation may involve a Negotiation Phase to procure an appropriate license. Such Negotiation is controlled by a subsystem of AXMEDIS workflow called Negotiation Workflow. In this section we deal with the AXEPTool Publication Scenario.

It is assumed that the publications tasks are normally carried out asynchronously and autonomously, without the intervention of the user. Moreover, the workflow instance contains the Task for uploading of the AXMEDIS Object on the sender's side and downloading of the AXMEDIS Object on the receiver's. The Scenario Proceeds as follows:

The workflow system is up and running.

We can 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.

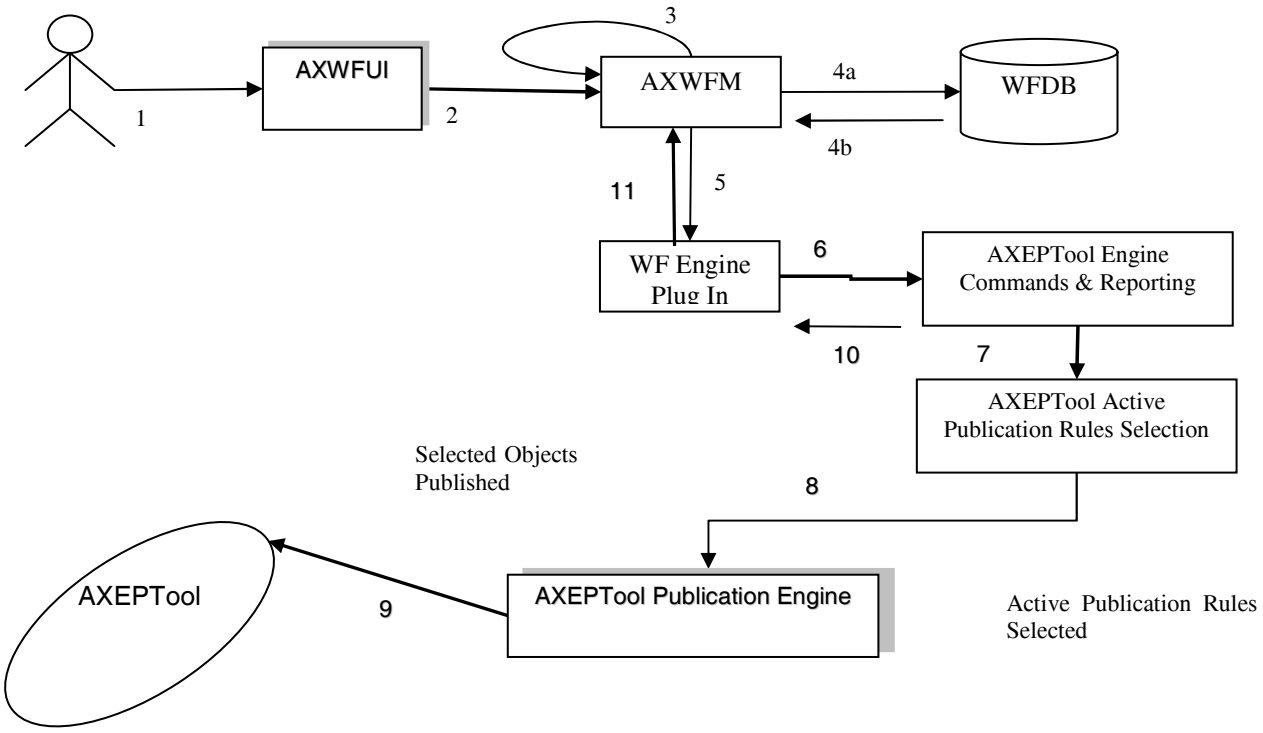
The workflow system passes the so-called Active Publication Request via the workflow plug-in to the AXEPTool Command & Reporting module to trigger the AXEPTool Active Publications Rule Selection Module which enables the selection and submission of appropriate objects for publication. The publication engine uses the thus activated publication request together with a Rule_ID to control the publication of the right object(s) from the Active List.

If the AXEPTool is not up and running, it is launched by the workflow system using the appropriate interfaces.

The AXEPTool Publication engine then moves the relevant AXMEDIS Object(s) to the "AXEPTool Out AXMEDIS Database Area" for publication on the P2P network under the control of the specified Rule_ID.

Upon completion of the activity, the AXEXPTool Publication engine informs the AXWF via the AXEPTool Comand & Reporting Module about the completion of the process, so it can proceed with the next step in the workflow-instance flow.

Scenario 3: Invoking the AXEPTool (Publish)



Scenario 4: Invoking the AXEPTool (Load)

This scenario describes the interaction between the workflow and the AXEPTool to share any AXMEDIS Object over the P2P network. There are two possible interaction scenarios between the workflow and the AXEPTool. On the one side, the interaction can be for uploading (Publishing) of some AXMEDIS Object(s), while on the other side the interaction can be for downloading (Loading) of the AXMEDIS Object. The Loading operation may involve a Negotiation Phase to procure an appropriate license. Such Negotiation is controlled by a subsystem of AXMEDIS workflow called Negotiation Workflow. In this section we deal with a Loading operation not requiring the invocation of the Negotiation workflow for License Procurement. The scenario proceeds as follows:

It is assumed that the user is interaction with the Workflow Management System.
Thus the user (Worker) is already authenticated and logged into the system and the workflow system is up and running.

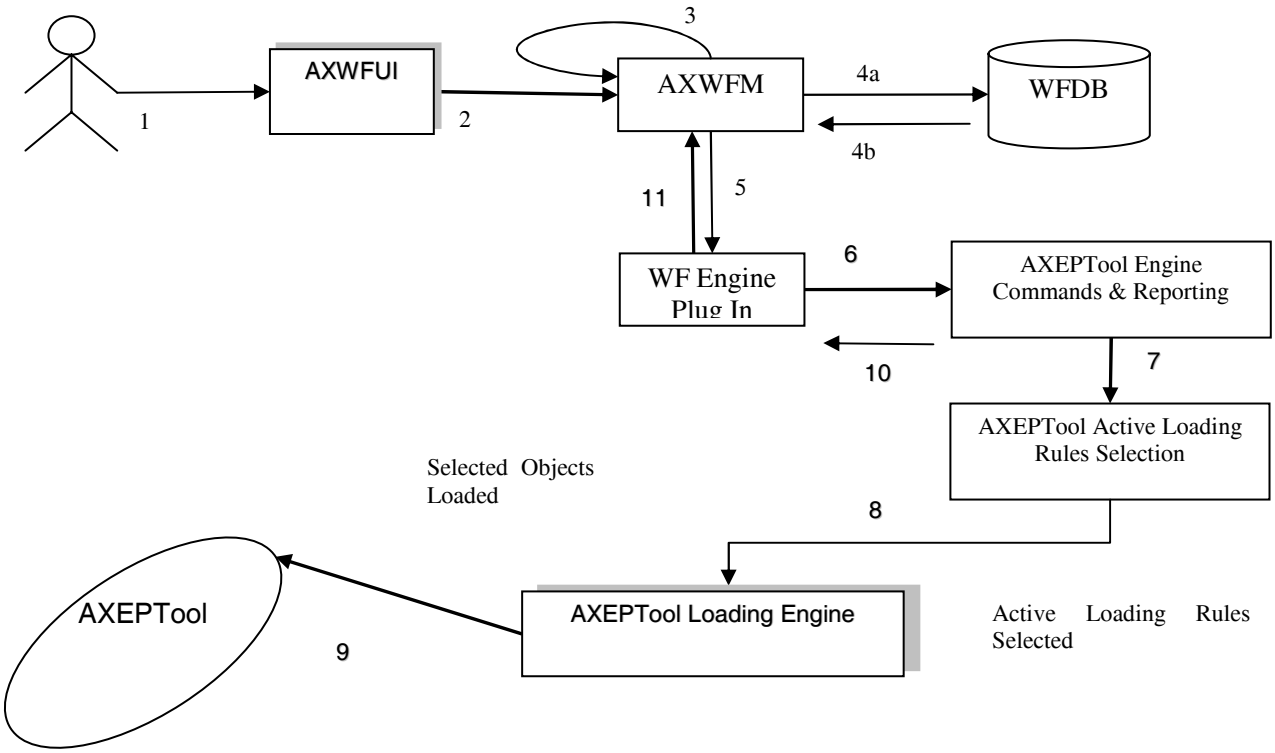
We can also assume that the client's (i.e. session-owner's) credentials_ID has been made available to be authenticated for sign-on to initiate the required exchanges information for authentication and billing purposes.

If any Task requires downloading of an AXMEDIS Object from the P2P network, the workflow system passes this request via the workflow plug-in to the AXEPTool Command & Reporting module to trigger the AXEPTool Active Loading Rule Selection Module which enables the selection and downloading of appropriate objects. The Loading engine uses the thus activated loading request together with a Rule_ID to control the downloading of the right object(s) from the Active List.

As soon as the required object is available on the P2P network, the Loading Tool Engine of AXEPTool downloads this AXMEDIS Object and moves it to the "AXEPTool In AXMEDIS Database Area".

When the AXEPTool Loading engine has completed the transfer, it informs the workflow system via the AXEPTool Command & Reporting module and the WFMS then moves this component to the appropriate location and proceeds to enable further tasks that could be perfumed once the object has become available.

Scenario 4: Invoking the AXEPTool (Loading)



Scenario 5: Invoking the AXEPTool (Load Upon Completion of Negotiation)

This scenario describes the interaction between the workflow and the AXEPTool for downloading (Loading) of an AXMEDIS Object when such Loading requires Negotiation as controlled by a subsystem of AXMEDIS workflow called Negotiation Workflow. In this section we deal with a Loading operation requiring the invocation of the Negotiation workflow for License Procurement. The Scenario proceeds as follows:

It is assumed that the user is interaction with the Workflow.

We can also assume that the client's (i.e. user/session-owner's) credentials_ID has been made available to be authenticated for sign-on to initiate the required exchange information for authentication and billing purposes.

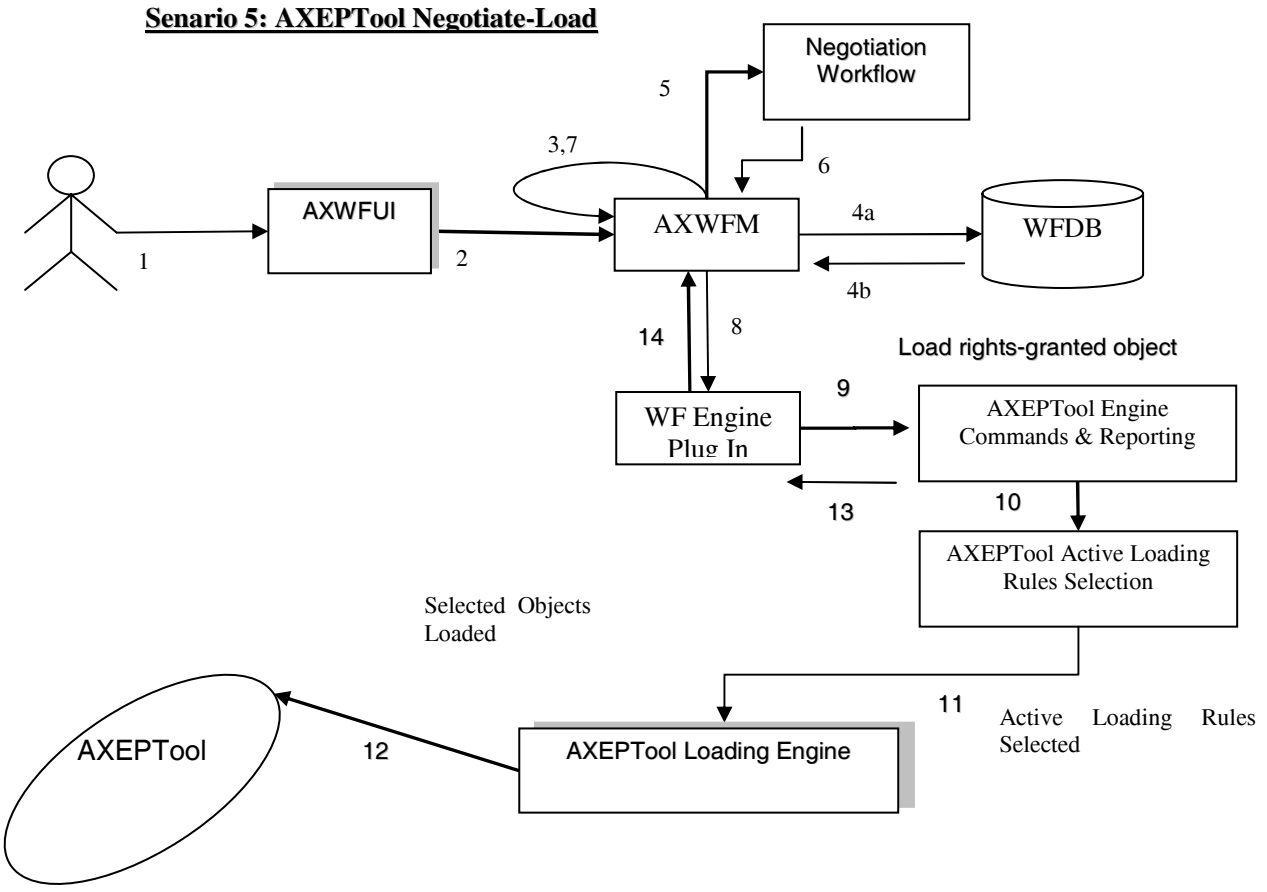
Thus the user (Worker) is already authenticated and logged into the system and the workflow system is up and running. It is assumed that the user is in interaction with the Workflow System and that the WFMS contains the AXMEDIS Object License Procurement Negotiation Workflow.

Suppose there is a Task that requires the downloading of an AXMEDIS Object from the P2P network, with the added complication that the user is required to enter into and complete a Negotiation Phase regarding the License Procurement of a particular AXMEDIS Object.

The workflow system passes the relevant Object_ID through to the AXMEDIS Query Support User Interface to set up an interaction with the AXMEDIS Object Manager (AXOM). The relevant Object Licensing particulars that thus become available are then passed to the License Procurement Negotiation Workflow to trigger the start of the Negotiation Phase.

Once the Negotiation Phase is completed the Object_ID is passed to the AXEPTool Loading Engine as usual, using the WF Plug-in, through the AXEPTool Command and Reporting module which enables a link to the AXEPTool Active Loading Rule Selection list. Once the selection of the relevant rule(s) for the download of the object is completed, then, as soon as the required Object becomes available on the P2P network, the Loading Tool Engine of AXEPTool downloads this AXMEDIS Object and moves it to the "AXEPTool In the AXMEDIS Database Area.

When the AXEPTool Loading engine has completed the transfer, it informs the workflow system via the AXEPTool Command & Reporting module and the WFMS then moves this component to the appropriate location and proceeds to enable further tasks that could be performed once the object download has been completed.



Scenario 6: 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:

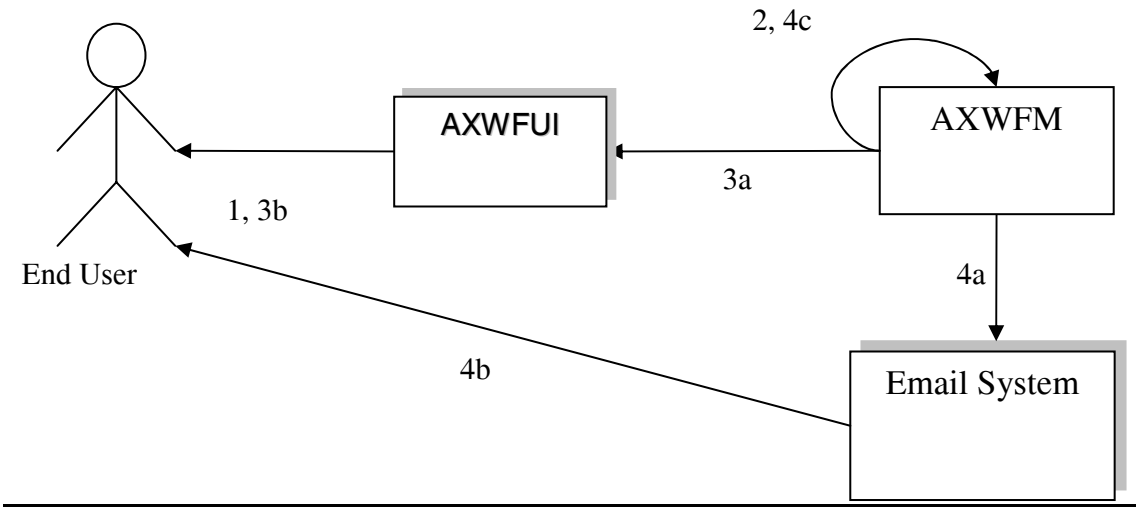
The user (Worker) is already authenticated and logged into the system and the workflow system is up and running. We can thus also assume that the client's (i.e. session-owner's) credentials_ID has been made available to be authenticated for sign-on to initiate the required exchanges information for authentication 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 7: 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.

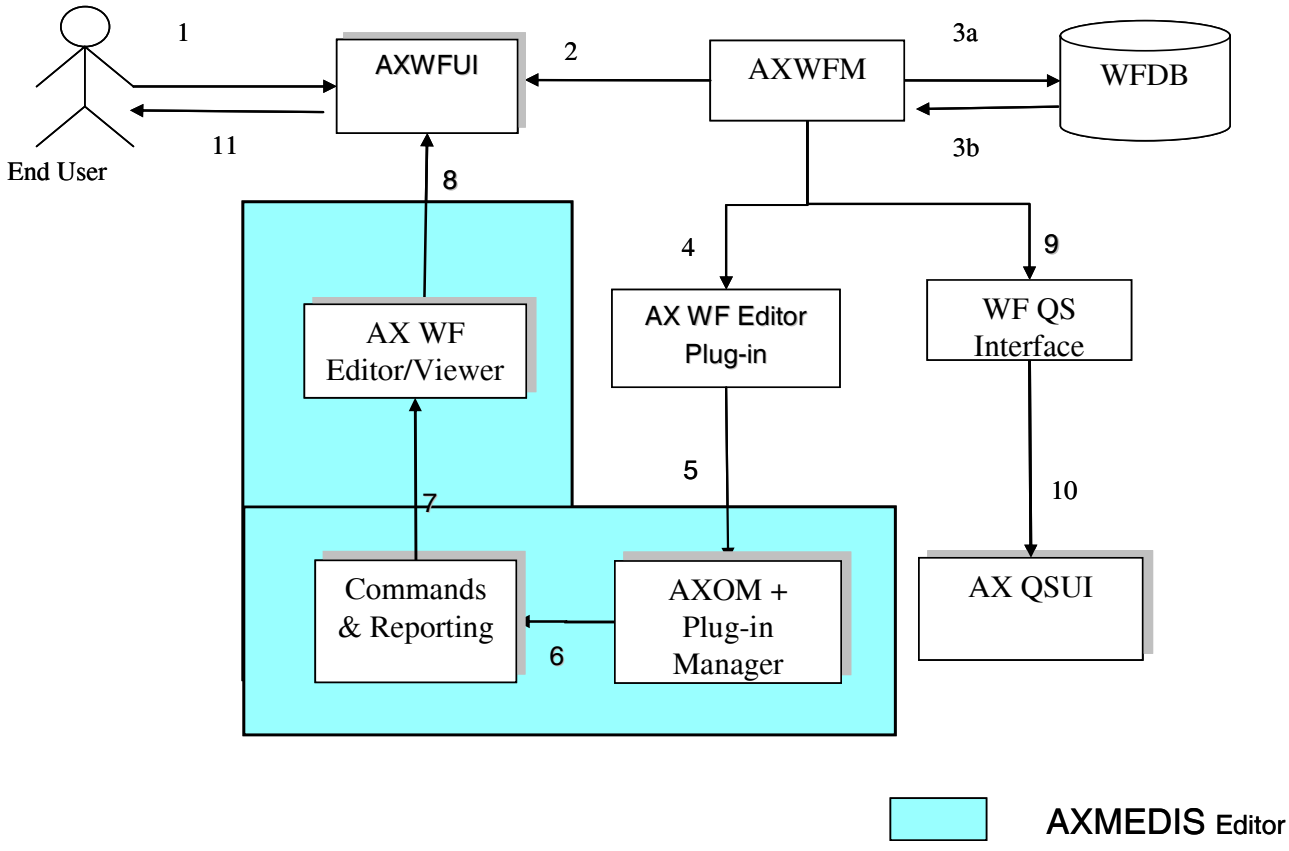
The Workflow system identifies all the components for the selected NPD and launches a set of queries to retrieve information for all of such components from the AXOM through the AXMEDIS Query Support Interface.

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 8: 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).

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.

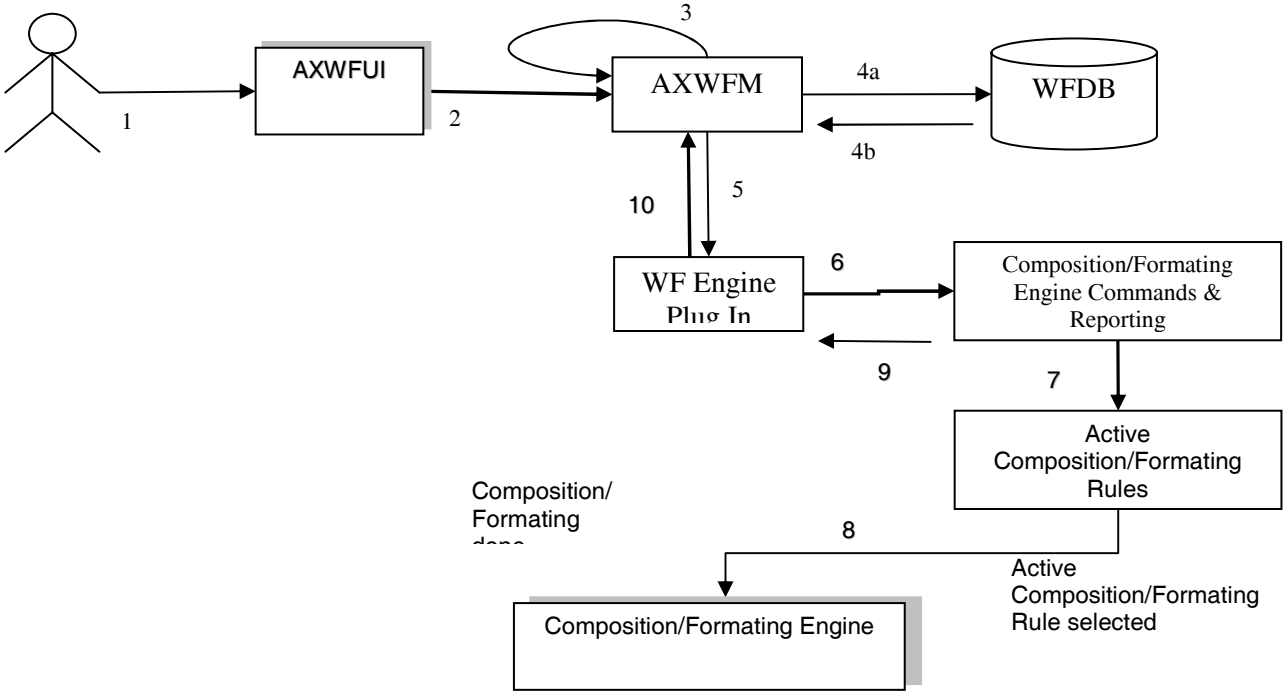
The workflow system is up and running.

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



6.2 Controlling and supervising local AXMEDIS tools

6.2.1 General WorkFlow Use Cases

6.2.1.1 Create NPD Workspace

UCId	UC6.2.1.1
Use case	Create a NPD
Description	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
Actors	Creator
Assumptions	Always valid: user has been identified by System; User has the correct rights
Steps	1 Actor chooses "Create NPD".
Post-conditions	New NPD project(s) space created in the user client & P2P desktops New NPD creation process instance started
Variations	Actor has no rights
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.2 Add components to the NPD

UCId	UC6.2.1.2
Use case	Add components to the NPD
Description	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
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	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
Post-conditions	New component added to active NPD. Started (if any) a subprocess for managing the newly created object
Variations	Actor has no rights Component and AXMEDIS Object incompatibility
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.3 Edit information of the NPD

UCId	UC6.2.1.3
Use case	Edit information of the NPD Note: this is a Use case with Workflow tight integration to editors (multiple interface)
Description	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.

Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	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.
Post-conditions	Proper editor invoked for active NPD.
Variations	Actor has no rights
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.4 Delete information of a NPD

UCId	UC6.2.1.4
Use case	Remove information of a NPD
Description	This is a generic use case responsible for removing anything from the NPD. e.g. partners, people, processes, components, etc.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	1 Actor selects component to remove then Actor clicks on 'remove'. 2 Optional confirmation dialogue.
Post-conditions	Selected component deleted from active NPD.
Variations	Actor has no rights
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.5 Show information regarding components of a NPD

UCId	UC6.2.1.5
Use case	Show information regarding components of a NPD
Description	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.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	1 Actor clicks on "Show". 2 An appropriate viewer is launched to show the requested details.
Post-conditions	Properties related to the active NPD displayed.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.6 Delete a NPD

UCId	UC6.2.1.6
Use case	Delete NPD
Description	This destroys the NPD workspace, when the decision of No-Go is taken. This removes all the information regarding the NPD.
Actors	Creator
Assumptions	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
Steps	1 Actor clicks on "Discard NPD". 2 Confirmation dialogue.
Post-conditions	Active NPD deleted along with associated components. The process instance initiated with the NPD instance creation is aborted.
Variations	No rights.
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.7 Search a NPD

UCId	UC6.2.1.7
Use case	Search a NPD
Description	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.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	1 Actor clicks on "Search" button. 2 A query dialogue box appears 3 Actor inserts the required parameters and launch the search
Post-conditions	The result of performed search is displayed in the form of a list
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.8 Track Component

UCId	UC6.2.1.8
Use case	Track component
Description	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).
Actors	Creator
Assumptions	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.
Steps	1 Actor selects a component. 2 Actor clicks on "Track component" button.

Post-conditions	History and planned steps of selected component displayed.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.9 Identify the CPA for a NPD

UCId	UC6.2.1.9
Use case	Track CPA
Description	This use case identifies the Critical Path Activities (CPA) for a NPD and produces all the information regarding those activities e.g. people involved, components being worked on, processes needing attention, etc.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	1 Actor clicks on "Identify CPA" button.
Post-conditions	Displays the critical path activities for the active NPD.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.10 Timestamp Generator

UCId	UC6.2.1.10
Use case	Timestamp generator
Description	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)
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	A non-empty NPD must be active/open
Steps	1 The workflow manager will log the beginning and end of any task performed on any object.
Post-conditions	An appropriate timestamp is associated with the objects.
Variations	This use case can be tested as expected result for each of the other cases.
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.11 Generate Versions

UCId	UC6.2.1.11

Use case	Generate version
Description	This generates hierarchical versions for all the digital and hard copy artefacts for the NPD development. This includes all versioning information to enable unique referencing of digital assets at any level of complexity e.g. work item, objects, workspace id, package id, bundle id, etc. i.e. both composite and single objects as well as work-in-progress objects and the workspaces in which they are being worked on.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	1 The workflow manager will generate versions of the AXMEDIS objects, including temporary versions, whenever necessary.
Post-conditions	New version of active component added to active NPD.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	Versions are simple identifiers necessary to distinguish objects. The objects can be the variations of same parent object or a temporary object created on the fly. may be the versioning is also stored into the workflow in terms of actions performed…

6.2.1.12 List of Work

UCId	UC6.2.1.12
Use case	List Work
Description	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).
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	Actor has been identified by System; There are work items to which the actor is assigned
Steps	1 Actor clicks on “show work list” button. 2 The actor can interact with the list of works
Post-conditions	The actor work list is displayed
Variations	
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.13 Select a Work Item from the List of Work

UCId	UC6.2.1.13
Use case	Select a Work Item from the List of Work
Description	This use case is responsible for selecting a work item from the work list
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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

Steps	1 Actor clicks on "select work item" button. 2 Once selected the actor can do some editing activities
Post-conditions	The actor work item activity list and/or description is displayed
Variations	
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.14 Complete a task of a work Item

UCId	UC6.2.1.14
Use case	Complete a task of a work Item
Description	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)
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	1 Actor clicks on "terminate work item task" button.
Post-conditions	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
Variations	
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.15 Distribute the assigned Work to process and people

UCId	UC6.2.1.15
Use case	Distribute the assigned Work to process and people
Description	This use case is responsible for distributing the work amongst the people assigned to the NPD. The work can be at component level or at NPD level. Some of the assigned work may be pipelined or suspended in a wait/pending stack, awaiting appropriate triggers for handover
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	1 Actor either selects a specific component first (to distribute at component level), or directly the Actor clicks on "Distribute work" button to distribute at the NPD level.
Post-conditions	Work is (re)scheduled for the selected component or NPD.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.16 Change State/Phase of a Task for a work Item

UCId	UC6.2.1.16
Use case	Change State/Phase of a Task for a work Item
Description	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.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	1 Actor selects a component 2 Actor then click "change state".
Post-conditions	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.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.17 Notification of information to a personnel for a task of a work

UCId	UC6.2.1.17
Use case	Notification of information to a personnel for a task of a work
Description	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.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	An non-empty NPD must be active/open
Steps	1 Actor selects one or more actors, select from list of message types then clicks "notify".
Post-conditions	Appropriate notification is sent to responsible actors via appropriate tool (e.g. email).
Variations	The actor may wish to type his own personal message.
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.18 Global Viewer of all information of a NPD

UCId	UC6.2.1.18
Use case	Global viewer of all information of a NPD
Description	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)
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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

Steps	1 Actor selects "global view".
Post-conditions	Global information is displayed/exported for the active NPD.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.19 Check-in task performed by manual operator

UCId	UC6.2.1.19
Use case	Check-in task performed by manual operator Note: this is a Use case with Workflow loose integration to editors (simple interface)
Description	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.
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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
Steps	1 Actor clicks on "check-in" button.
Post-conditions	The object is copied from AXMEDIS DB To an area for exclusive access of the actor, ready to be downloaded
Variations	Actor has no rights
Asynchronous actions	None
Design suggestions	None
Issues	None

6.2.1.20 Check-out task performed by manual operator

UCId	UC6.2.1.20
Use case	Check-out task performed by manual operator Note: this is a Use case with Workflow loose integration to editors (simple interface)
Description	This use case is responsible for copying the object from the actor exclusive access area (when he previously uploaded it) to the AXMEDIS DB
Actors	Creator, Producer, Integrator, Aggregator
Assumptions	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.
Steps	1 Actor clicks on "check-out" button.
Post-conditions	The file is copied in the AXMEDIS DB
Variations	Actor has no rights It can automatically execute the "task completed"
Asynchronous actions	None
Design suggestions	None
Issues	None

6.3 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

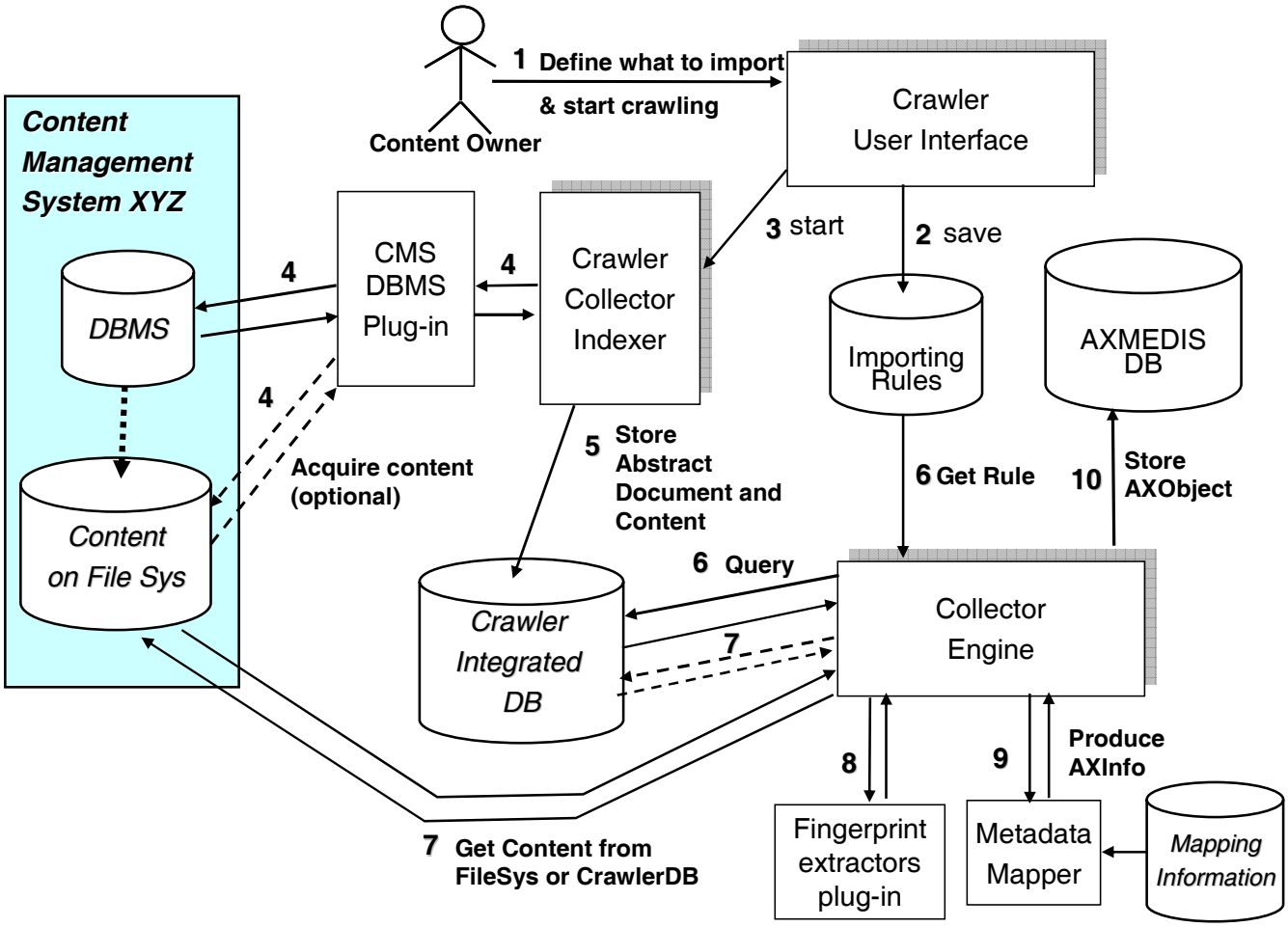
b) Create/modify objects or object-parts (e.g. new required views, new rendering, new modifications on embedded element(s) etc.)

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.

7 AXMEDIS Object Acquisition from CMS



7.1 Automatic gathering of Content, Collector Engine

7.1.1 Setup for metadata mapping

UCId	UC7.1.1
Use case	Setup for metadata mapping
Description	An Actor wants to setup the environment for CMS crawling.
Actors	Content Provider, Content Distributor
Assumptions	None
Steps	1 The Actor, defines how to map metadata coming from the CMS to the AXMEDIS model.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	An XLS may be used for this task
Issues	None

7.1.2 Setup for content crawling

UCId	UC7.1.2

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

7.1.3 Define what content to acquire from Crawled Integrated Database

UCId	UC7.1.3
Use case	Define what content to acquire from Crawled Integrated Database
Description	An Actor wants to setup the environment for CMS gatering.
Actors	Content Provider, Content Distributor
Assumptions	Content crawling environment was set up.
Steps	1 The Actor, interacts with the Collector Engine User Interface to add/remove/modify an importing rule. 2 When adding/modifying a rule the Actor states: 2.1 which properties has to have the content to be imported (e.g. all audio files of Ramazzotti, later that 1990)
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

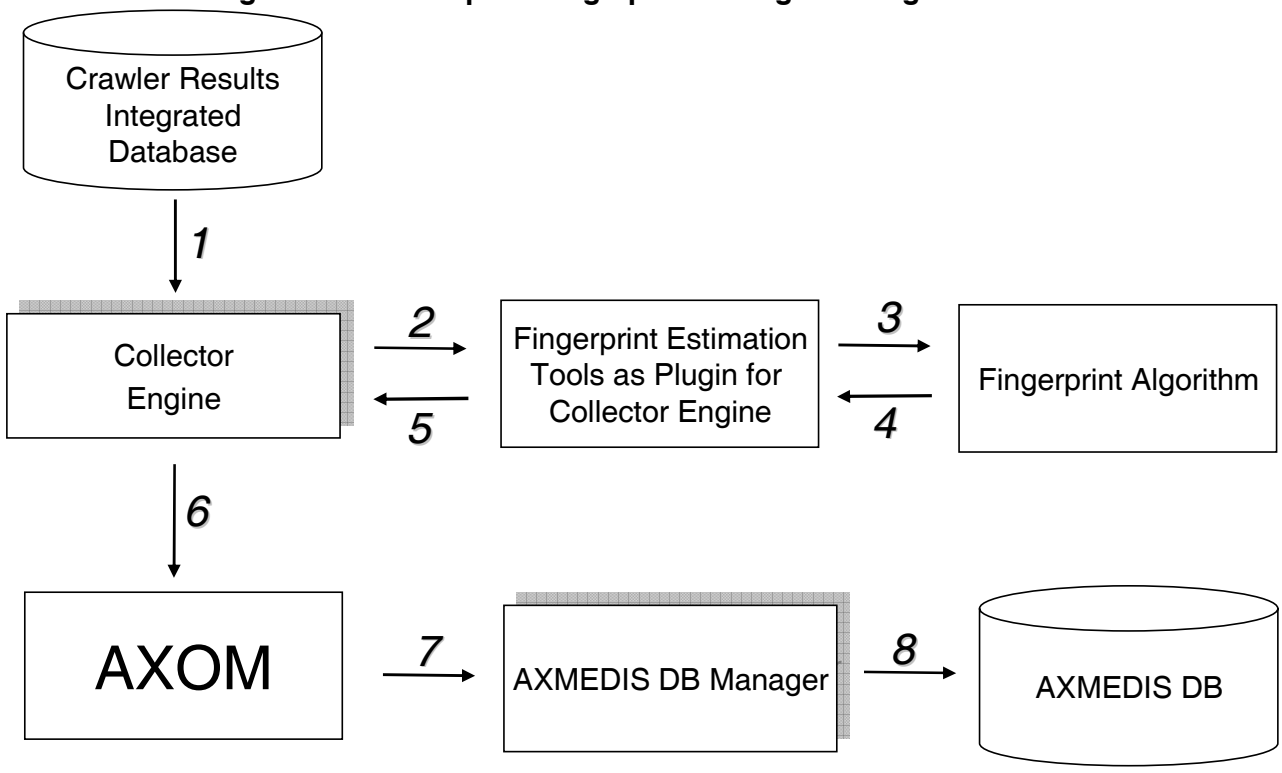
7.1.4 Start content crawling

UCId	UC7.1.4
Use case	Start content crawling
Description	An Actor wants to start CMS crawling.
Actors	Content Provider, Content Distributor
Assumptions	The metadata mapping and the crawler have been set up.

Steps	1 The Actor, starts CMS crawling using Collector Engine User Interface 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. 2.1 the Collector Engine accesses to the new/updated content and checks the import rules 2.2 If the content has to be imported 2.2.1 the content is fingerprinted and/or adapted (using plug-ins) as instructed by the import rule 2.2.2 the AXMEDIS Object is built using the metadata coming from the Crawler Database 2.2.3 the AXMEDIS Object is uploaded in the AXMEDIS Database
Post-conditions	None
Variations	None
Asynchronous actions	Content Acquisition can be stopped
Design suggestions	None
Issues	None

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

7.2.1 Calculating content descriptors/fingerprint during crawling



UCId	UC7.2.1
Use case	Calculating content descriptors/fingerprint during crawling
Description	After collecting content by the crawler collector indexer the crawler collector indexer initiates the calculation of the fingerprint.
Actors	Producer

Assumptions	Content was indexed by Crawler Collector Indexer
Steps	1 Fingerprinting method is called by Crawler Content Indexer 2 Content descriptors and fingerprint/descriptors values are calculated. 3 Content descriptors and fingerprint/descriptors values are stored in the database
Post-conditions	Content descriptors and fingerprints are stored in the database and into the AXMEDIS object structure to save them into the AXOB.
Variations	• Different kinds of media types • Compound objects
Asynchronous actions	
Design suggestions	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).
Issues	In general, content descriptors are 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 cryptographical hash functions have to be considered.

8 AXMEDIS Database

8.1 Managing a Database of AXMEDIS Objects

8.1.1 Administer Objects in the AXMEDIS DB

UCId	UC8.1.1
Use case	Administer Objects in the AXMEDIS DB
Description	An Actor performs administrative tasks on AXMEDIS DB related to objects.
Actors	Content Integrator, Content Distributor, and in general all the user that have an AXMEDIS DB in-house
Assumptions	None
Steps	2 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 3 The AXMEDIS object database perform the operations 4 The actor is notified about the success of the operation and the result of the operation id transmitted back to the actor.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

8.1.2 Administer User in the AXMEDIS DB

UCId	UC8.1.2
Use case	Administer Objects in the AXMEDIS DB
Description	An Actor perform administrative tasks on AXMEDIS DB related to users.
Actors	Content Integrator, Content Distributor, and in general all the user that have an AXMEDIS DB in-house
Assumptions	None
Steps	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 2 The AXMEDIS object database perform the operations 3 The actor is notified about the success of the operation and the result of the operation id transmitted back to the Actor.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

8.1.3 Accessing a specific version of an AXMEDIS object

UCId	UC8.1.3
Use case	Accessing a specific version of an AXMEDIS object
Description	An Actor wants to see a specific version of an object interacting with the AXMEDIS Database Administration Tool.
Actors	Content Integrator, Content Distributor, Content Owner

Assumptions	None
Steps	1 The Actor, login into the system specifying the username and a password, if not already done 2 The Actor makes a query or browses the database looking for the object 3 The Actor selects to see the history of the object 4 The Tool reports the history containing the date of upload, the name of the user who performed the upload, a brief description of the reason of new version 5 The Actor selects the version to view and opens it using the AXMEDIS Editor
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	When a specific version is opened using the AXMEDIS Editor, AXMEDIS Editor cannot upload it again in the database

8.1.4 Removing last version of an AXMEDIS object

UCId	UC8.1.4
Use case	Removing last version of an AXMEDIS object
Description	An Actor wants to remove the last version of a specific object interacting with the AXMEDIS Database Administration Tool.
Actors	Content Integrator, Content Distributor, Content Owner
Assumptions	None
Steps	1 The Actor, login into the system specifying the username and a password, if not already done 2 The Actor makes a query or browses the database looking for the object 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.
Post-conditions	None
Variations	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. The Actor may ask to remove versions to reconstruct the state of the object/objects up to a specific date.
Asynchronous actions	In case of multiple removal the Actor can stop the process.
Design suggestions	None
Issues	None

8.1.5 Removing an AXMEDIS object

UCId	UC8.1.5
Use case	Removing an AXMEDIS object
Description	An Actor wants to remove a specific object interacting with the AXMEDIS Database Administration Tool.
Actors	Content Integrator, Content Distributor, Content Owner
Assumptions	None

Steps	1 The Actor, login into the system specifying the username and a password, if not already done 2 The Actor makes a query or browses the database looking for the object 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.
Post-conditions	None
Variations	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.
Asynchronous actions	In case of multiple removal the Actor can stop the process.
Design suggestions	None
Issues	If transactions have been performed for the object the operation may fail

8.1.6 User Management

UCId	UC8.1.6
Use case	User Management
Description	An Actor wants to add/remove/modify user information to access to the database
Actors	Content Integrator, Content Distributor, Content Owner
Assumptions	None
Steps	1 The Actor, login into the system specifying the username and a password, if not already done 2 The Actor selects the user management tool 3 The Tool presents the list of users having access to the system (if the user has the user management grant) 4 The Actor can: 4.1 Add a new user giving: the username, a password, e-mail, description, the groups he/she belongs to, the grants 4.2 Remove an user 4.3 Modify the user data
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	The OS or DBMS users management support may be used.
Issues	Users should be kept synchronized with the workflow users

8.1.7 User Groups Management

UCId	UC8.1.7
Use case	User Groups Management
Description	An Actor wants to add/remove/modify groups of users
Actors	Content Integrator, Content Distributor, Content Owner
Assumptions	None

Steps	1 The Actor, login into the system specifying the username and a password, if not already done 2 The Actor selects the group management tool 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 4 The Actor can: 4.1 Add a new group 4.2 Remove a group (if no users are in it) 4.3 Modify the group 4.4 Add/remove users from the groups
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	The OS or DBMS users management support may be used.
Issues	Users should be kept synchronized with the workflow users

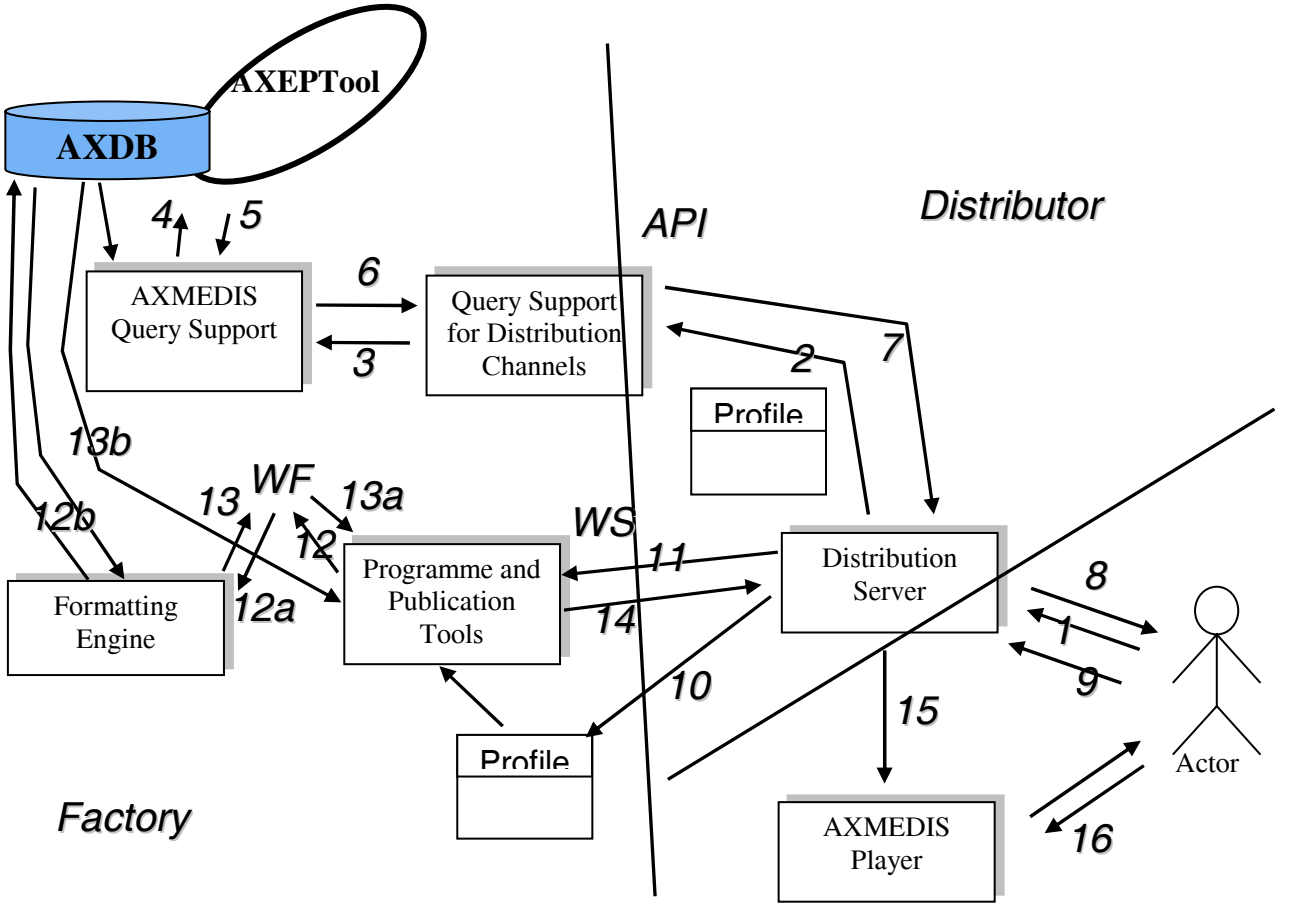
8.2 Making queries inside Databases of AXMEDIS objects and inside the objects

Selection creation is explained in the general use case section.

8.2.1 Querying for AXMEDIS objects and inside objects

UCId	UC8.2.1
Use case	Querying for AXMEDIS objects and inside objects
Description	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.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	None
Steps	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. 2 The Actor submits the queries previously composed to execute the search 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 4 AXMEDIS Query Support merges the results all together and return the complete list to the AXMEDIS Query Support User Interface 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…
Post-conditions	None
Variations	
Asynchronous actions	None
Design suggestions	Web services should be adopted at the communication layer
Issues	None

8.2.2 Querying for AXMEDIS from Clients



UCId	UC8.2.2
Use case	Querying for AXMEDIS From Clients
Description	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.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	None

Steps	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. 2 The Actor submits the queries previously composed to execute the search. 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 4 AXMEDIS Query Support merges the results all together and return the complete list to the AXMEDIS Query Support for Clients. 5 AXMEDIS Query Support for Clients trnascodes 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… 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. 7 The Programme and Publication Tool Engine initiates a work flow (WF) for the formatting of the content. 8 After finishing the WF the object is sent to the distribution server, which forwards it to the requesting Actor.
Post-conditions	None
Variations	Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	Web services should be adopted at the communication layer
Issues	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.

8.2.3 Bookmark a query

UCId	UC8.2.3
Use case	Bookmark a query
Description	An Actor can save an executed query for future reuse.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	The bookmarked query must already be executed
Steps	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 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.
Post-conditions	None
Variations	Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	None
Issues	None

8.2.4 Retrieve a bookmarked query

UCId	UC8.2.4
Use case	Retrieve a bookmarked query
Description	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.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	At least a query must exist in the bookmark of the user profile
Steps	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 2 The AXMEDIS Query Support User Interface show the query in the query environment for issuing that query of for modifying it
Post-conditions	None
Variations	Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	None
Issues	None

8.2.5 Organize bookmarked queries

UCId	UC8.2.5
Use case	Organize bookmarked queries
Description	An Actor can organize the queries in the bookmark dividing them in folders according her/his needs.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	None
Steps	1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, recall from his/her user profile the bookmarks 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 3 The Actor confirms the new configuration of the bookmarks 4 The AXMEDIS Query support save the new bookmark collection in the user profile
Post-conditions	None
Variations	3a. The Actor cancel the modifications Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	An user interface similar to that of Internet browser can be adopted.
Issues	None

8.2.6 Save an incomplete query

UCId	UC8.2.6
Use case	Save an incomplete query
Description	An Actor can save an incomplete or not executed query for future reuse.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	None
Steps	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 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.
Post-conditions	None

Variations	Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	None
Issues	None

8.2.7 Retrieve an incomplete query

UCId	UC8.2.7
Use case	Retrieve an incomplete query
Description	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.
Actors	Content Integrator, Content Distributor, Content Consumer
Assumptions	At least a query must exist in the user profile
Steps	1 The Actor, after performing authentication, by using the AXMEDIS Query Support User Interface, recall from the user profile a stored query 2 The AXMEDIS Query Support User Interface shows the query in the query environment for issuing that query of for modifying it
Post-conditions	None
Variations	Query Support for Clients (with reduced query functionality)
Asynchronous actions	None
Design suggestions	None
Issues	None

9 AXMEDIS AXEPTools for P2P distribution on B2B

9.1 AXEPTool for P2P on B2B

9.1.1 Discovery and connection of peers on B2B P2P network

UCId	UC_9.1.1
Use case	Discovery and connection of peers on B2B P2P network
Description	The user wants to discover and connect to one or more peers already connected on the P2P network.
Actors	The AXEPTool user.
Assumptions	The user launches the AXEPTool.
Steps	1 The user press the "Connect" button in the GUI 2 The AXEPTool starts a discovery protocol to find out on the network one or more participants in the AXMEDIS P2P network. 3 The AXEPTool receives a list of hosts enabled to accept incoming connections. 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). That implies the involvment of a external certification/supervisor authority. 5 Hosts without a certified identity causes the handshaking to fail. They ARE NOT allowed to join the AXMEDIS community. 6 The remote host can refuse the connection because is busy. Then other remote hosts are contacted 7 After a timeout, if no connection succeed the AXEPTool popup a "Unrecoverable Error" message to the user.
Post-conditions	If one or more connections succeed the user see the status "CONNECTED" in the status bar. Otherwise, a error message is popped-up When the connection is established the AXEPTool is ready to exchange messages with the other participants in the community.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

9.1.2 Report P2P downloads/uploads network traffic

UCId	UC_9.1.2
Use case	Report P2P downloads/uploads network traffic
Description	AXEPTool provides real-time, auto-refreshing, P2P network traffic reports for Downloads and Uploads. Moreover, uploads/downloads can be suspended, terminated, and resumed.
Actors	The AXEPTool user.
Assumptions	AXEPTool is connected to the network, some uploads/download are running .
Steps	1 The user opens the "Download/Upload Table" in the UI 2 Data regarding the messages exchanged are presented to the user 3 The user selects one upload/download and suspend/resume/terminate the selected session
Post-conditions	A download/upload is suspended/resumed/terminated
Variations	None
Asynchronous actions	None
Design suggestions	None

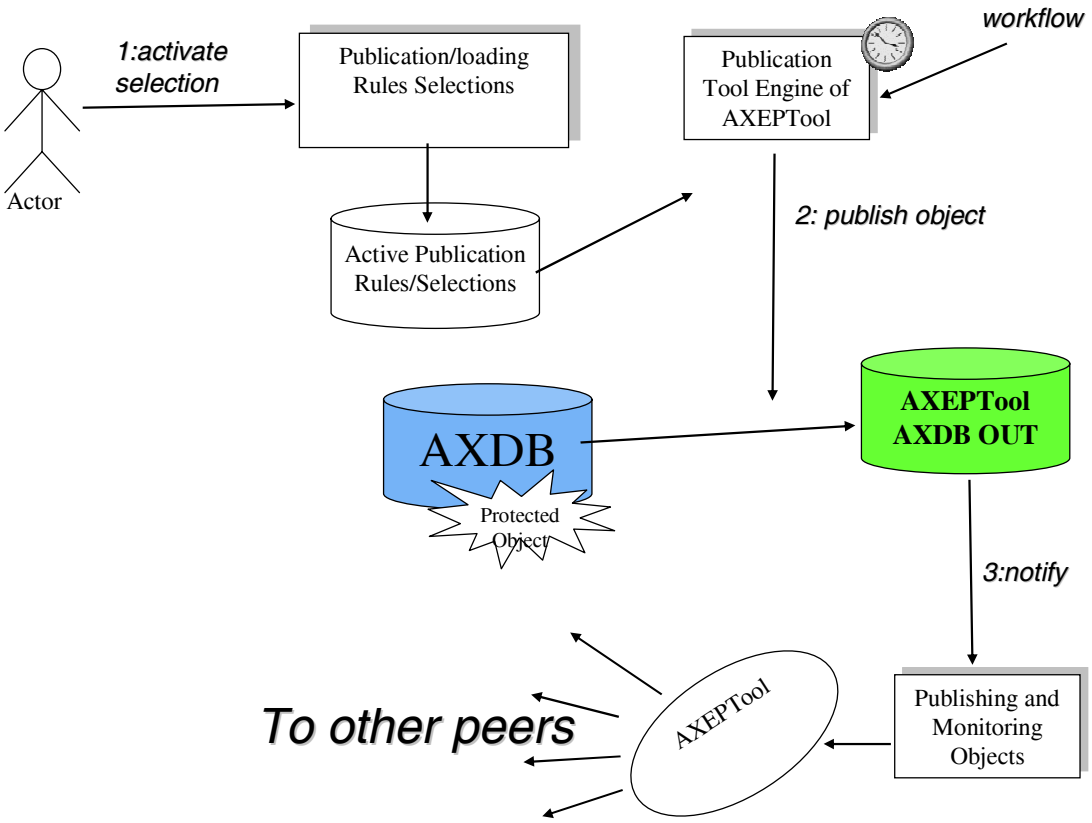
Issues	None

9.2 Publication and loading AXMEDIS Objects of AXEPTool

9.2.1 Creation of a publishing rule for the AXEPTool

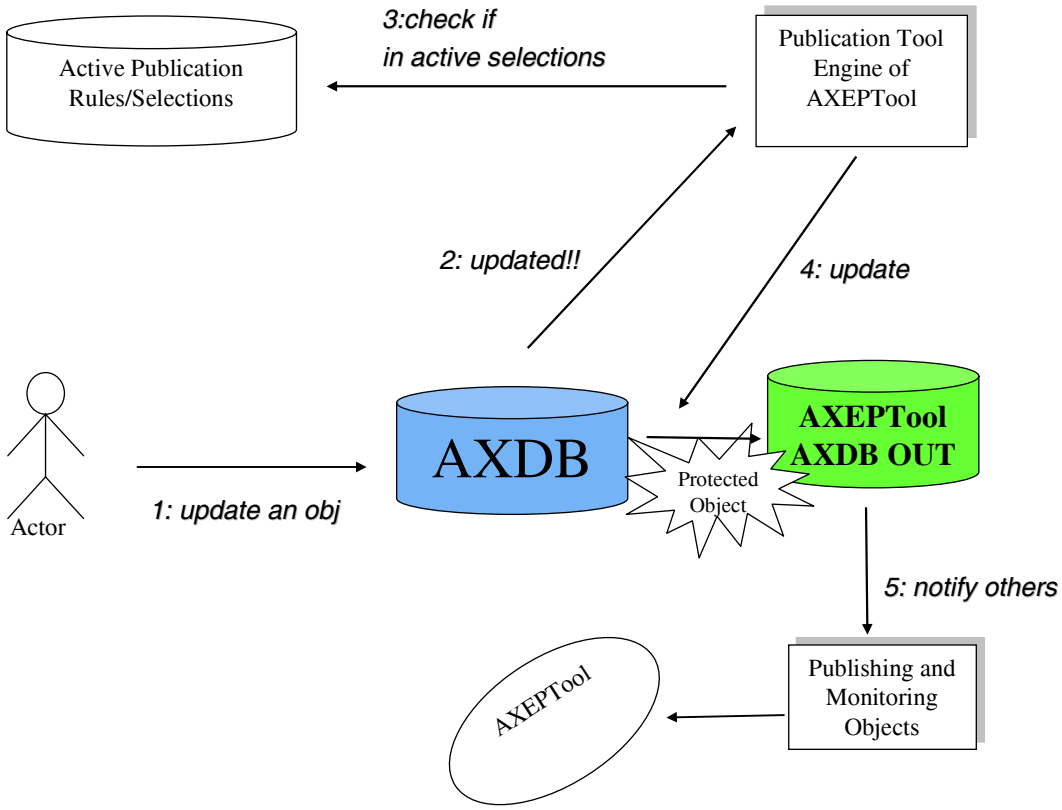
UCId	UC_9.2.1
Use case	Creation of a publishing rule for the AXEPTool
Description	The Publication Tool Engine allows the user to build publication rules in two ways: by example and by the Rule Editor User Interface
Actors	Content Owner.
Assumptions	One or more objects are stored in the AXMEDIS Data Base
Steps	1a) The user opens the Publication/Loading Rules/Selections of the Publication tool engine of AXEPTool 2a) The user fills the data required to build a new publication rule. Or alternatively 1b) The user manually selects an AXMEDIS object in the AXMEDIS Data Base, or select them as a result of a query and thus from a Selection. 2b) The user invokes the function "Build rule by example" in both cases 3. The Publication Tool Engine of AXEPTool saves the new rule in the AXEPTool Active Publication Rules/Selections
Post-conditions	A selection of AXMEDIS object is available on the P2P network.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

9.2.2 Automatic publication of a selection of objects on the AXEPTool



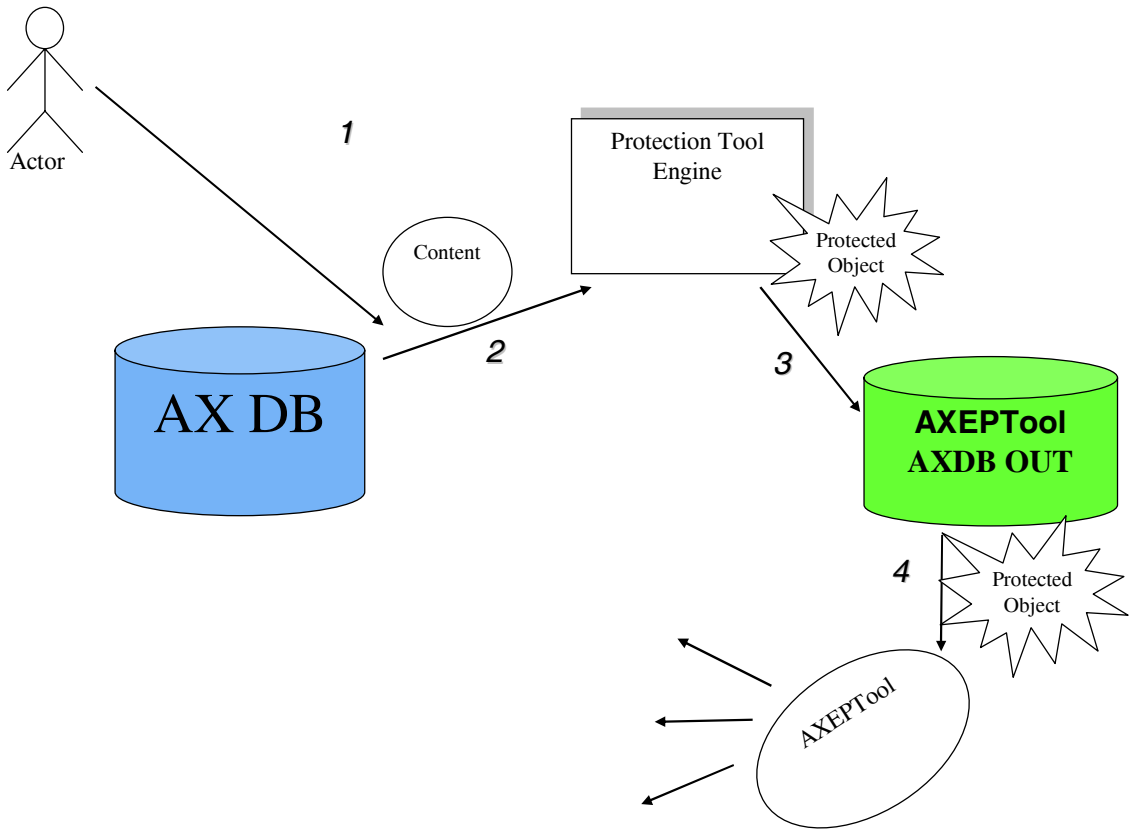
UCId	UC_9.2.2
Use case	Automatic publication of a selection of objects on the AXEPTool.
Description	The Actor wants to publish one o more AXOB on the AXEPT Tool network.
Actors	Aggregator, Producers
Assumptions	The user creates a Selection of one or more AXOB. The AXMEDIS Objects are stored in the AXDB.
Steps	1 The Actor, through the Publication/Loading Rules/Selections, submits the Selection to the AXEPTool Active Publication Active Rules/Selections o the Selection becomes active o the Actor is allowed to modify the default activation period 2 According to activation periods the Publication Tool Engine of AXEPTool publishes each objects of the Selection on the AXEPTool OUT AXDB. 3 The AXEPTool OUT AXDB advises the Publishing and Monitoring Objects which provides to broadcast the event to all its counterparts on the network
Post-conditions	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.
Variations	Publication of AXMEDIS objects on AXEPTool can be also made using the AXMEDIS Workflow Manager.
Asynchronous actions	None
Design suggestions	AXEPTool IN/OUT AXDB is an instance of the AXMEDIS database manager.
Issues	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.

9.2.3 Automatic updating of a modified object on the AXEPTool



UCId	UC_9.2.3
Use case	Automatic updating of a modified object on the AXEPTool.
Description	The user decides to activate the procedure of updating. The aim is to update an AXOB already published.
Actors	Aggregator, Producers
Assumptions	The AXOB is already been published. The AXOB has been modified in the AXDB by the owner.
Steps	1 The user activates the updating process. 2 The local AXDB advises the Publication Tool Engine of AXEPTool that the object has been updated. 3 The AXOB has to belong to one of the active Selections. 4 Publication Tool Engine of AXEPTool updates the AXOB contained into the AXEPTool OUT AXDB. 5 The other Peers in the network are notified.
Post-conditions	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.
Variations	None
Asynchronous actions	None
Design suggestions	AXEPTool IN/OUT AXDB is an instance of the AXMEDIS database manager.
Issues	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.

9.2.4 Automatic publication of a not protected object on the AXEPTool



UCId	UC_9.2.4
Use case	Automatic publication of a non protected object on the AXEPTool
Description	The user wants to publish a non protected object. The aim is to improve the starting content with the suitable protection information and eventually to publish it.
Actors	Aggregator, Producers
Assumptions	The unprotected content is stored in the AXDB.
Steps	1 The user starts the procedure of publishing. 2 The Protection Tools Engine gets the content from the AXDB and creates an AXOB with the proper protection part. 3 The Protection Tools Engine through the Publication Tool Engine publishes the AXOB on the AXEPTool OUT AXDB. 4 The event is broadcasted to all counterparts on the network.
Post-conditions	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.
Variations	None
Asynchronous actions	None
Design suggestions	AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
Issues	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.

9.2.5 Manual Publication of AXMEDIS Objects with the AXEPTool

UCId	UC_9.2.5
Use case	Manual Publication of AXMEDIS Objects with the AXEPTool
Description	The user wants to publish an AXMEDIS object in the AXEPTool. Objects are copied into the Output Database of the AXEPTool.
Actors	Content Owner.
Assumptions	One or more objects are stored in the AXMEDIS Data Base
Steps	1 The user invokes the Publication Tool Engine by the Publication Engine User Interface. 2 The user selects objects in the AXMEDIS Data Base. 3 If a selected object is not protected and the user wants it, the Publication Tool Engine invokes the Protection Tool Engine to protect the object. 4 Selected objects are copied in the AXEPTool OUT AXMEDIS Output Database
Post-conditions	A selection of AXMEDIS object is available on the P2P network.
Variations	None
Asynchronous actions	None
Design suggestions	AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
Issues	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.

9.2.6 Producing a query to search on the AXEPTool

UCId	UC_9.2.6
Use case	Producing a query to search on the AXEPTool
Description	The user wants to produce a query in order to search AXMEDIS objects on P2P network.
Actors	The AXEPTool user.
Assumptions	The peer is connected to the P2P network.
Steps	1 The user opens the Advanced Query UI to produce a technical query. 2 The user fill in the fields with the necessary information. 3 The user starts the query.
Post-conditions	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.
Variations	None
Asynchronous actions	The user can launch more than one query at a time.
Design suggestions	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
Issues	Manual browsing should be available as well.

9.2.7 View/Manage query results coming from the AXEPTool

UCId	UC_9.2.7
Use case	View/Manage query results coming from the AXEPTool
Description	The user wants to manage with the results of the querying process.
Actors	The AXEPTool user.
Assumptions	A query has been sent.

Steps	1 The user opens the query result sheet for the query he/she is interested. 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. 3 The user can sort/delete/make selections on the query result sheet.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	This can be unified with the Query Support. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
Issues	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.

9.2.8 Active query pool management for the AXEPTool

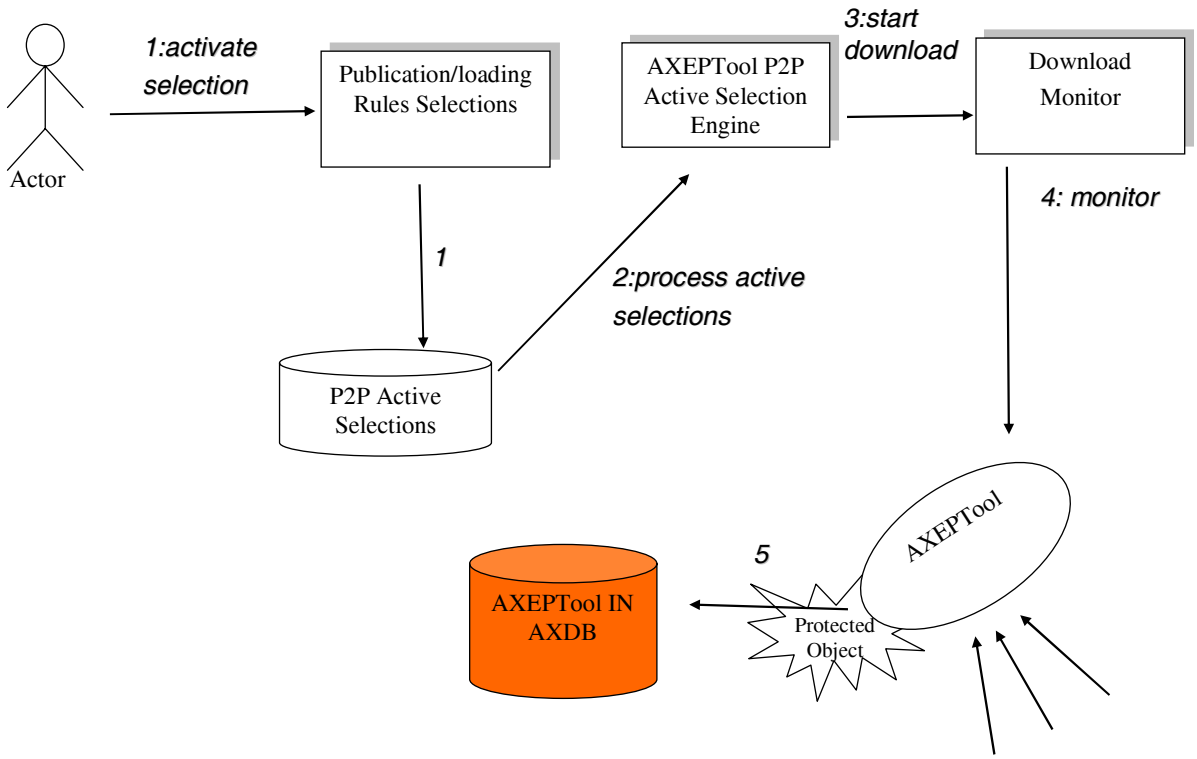
UCId	UC_9.2.8
Use case	Active query pool management for the AXEPTool
Description	The user wants to keep the AXEPTool up-to-date with respect to a particular query.
Actors	The AXEPTool user.
Assumptions	One or more queries have been sent. Their query result sheet is available in the GUI
Steps	1 The user selects one query result sheet. 2 The user select a time-interval for the query to be re-sent to the network
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	This can be unified with the Query Support. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
Issues	None

9.2.9 Downloading an AXMEDIS object

UCId	UC_9.2.9
Use case	Download AXMEDIS Object
Description	The user chooses to download a selection of AXMEDIS objects available on the P2P network.
Actors	The AXEPTool user.
Assumptions	One or more objects are shown as available in the P2P network within a query result sheet.
Steps	1 The Actor selects one or more objects in a query result sheet and starts the download . 2 AXEPTool verifies DRM rules, protections and licensing aspects. 3 A download session is started. A download session sheet is created and inserted in the GUI.
Post-conditions	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.
Variations	None

Asynchronous actions	The Actor can start, suspend, cancel or resume the download session of an object
Design suggestions	Feedback on download status must be implemented. AXEPTool IN AXDB is an instance of the AXMEDIS database manager.
Issues	none

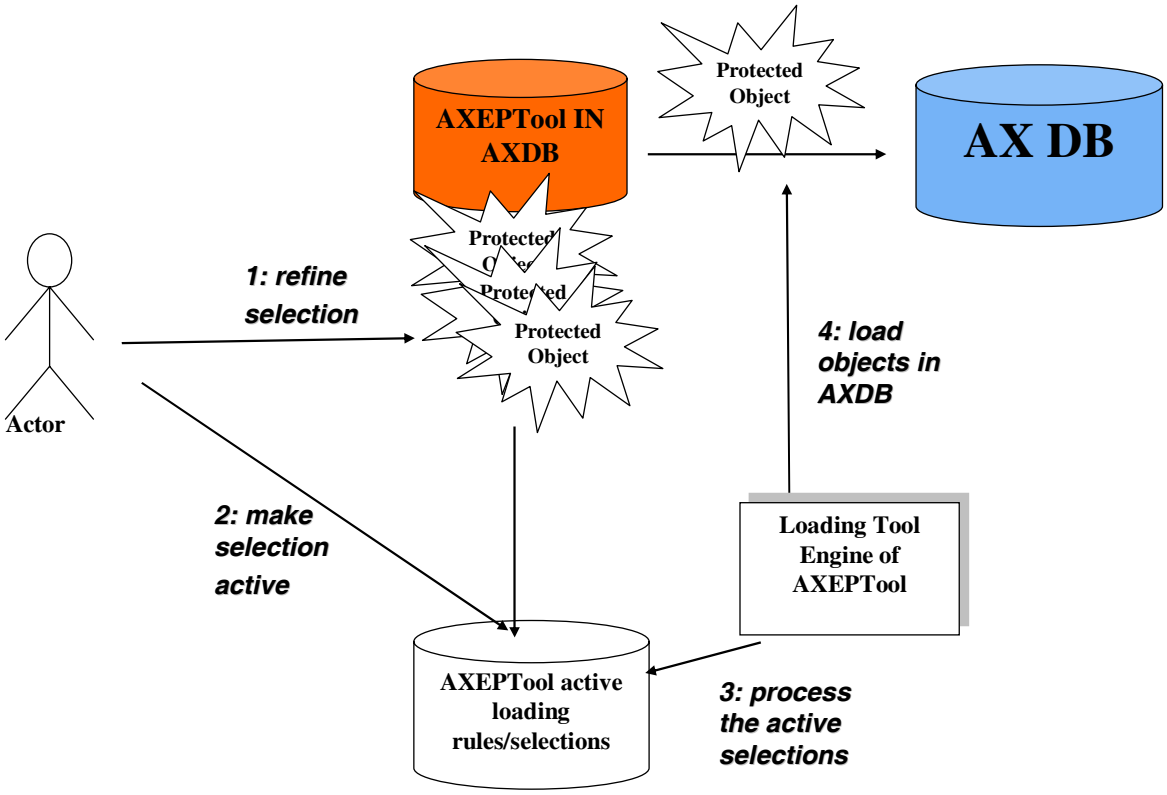
9.2.10 Automatic downloading of a selection of objects available in the P2P network



UCId	UC_9.2.10
Use case	Automatic loading of a selection of objects available in the P2P network.
Description	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.
Actors	Aggregator, Content Provider, Publisher
Assumptions	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.

Steps	1 The User activates the Selection by using the Publication/Loading Rules/Selections Editor. 2 AXEPTool P2P Active Selection Engine elaborates the active Selections contained in the P2P Active Selections. 3 AXEPTool P2P Active Selection Engine downloads each AXOB of the Selection. 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. 5 The object is stored in the AXEPTool IN AXDB.
Post-conditions	The Actor can play, run, visualize, etc, each object accordingly to the related DRM rules.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

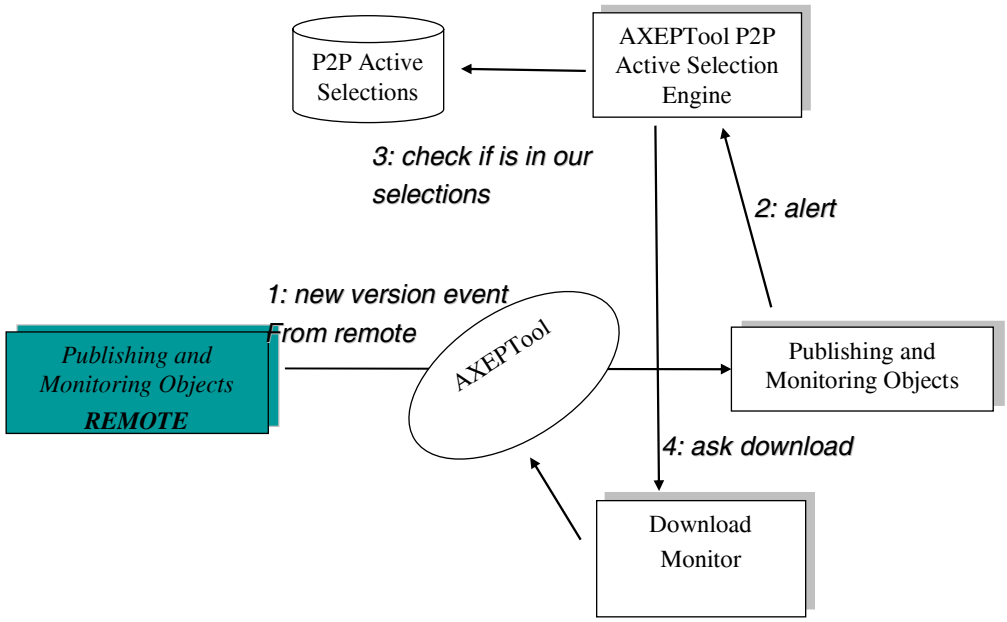
9.2.11 Refining the selection (Active Selections) for the AXEPTool



UCId	UC_9.2.11
Use case	Refining the selection (Active Selections) for the AXEPTool
Description	The Actor decides which AXOB is interested in.
Actors	Aggregator, Content Provider, Publisher
Assumptions	The Actor has tried the loaded objects, according to the related DRM rules.

Steps	1 the Actor selects only the AXOB he/she is interested in. 2 the Selection become Active by submitting it to the AXEPTool Active Loading Rules/Selections 3 the Loading Tool Engine of AXEPTool elaborates the Active Selection. 4 Each object of the selection is loaded into the local AXDB.
Post-conditions	The AXOB are in the AXDB.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

9.2.12 Automatic loading new versions of AXMEDIS Objects for the AXEPTool



UCId	UC_9.2.12
Use case	Automatic loading new versions of AXMEDIS Objects for the AXEPTool
Description	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.
Actors	None
Assumptions	One or more Active Selection have already been produced. A new version of a previously downloaded object is published.

Steps	1 Publication and Monitoring Objects is informed of the updating. 2 AXEPTool P2P Active Selection Engine is alerted by Publication and Monitoring Objects. 3 AXEPTool P2P Active Selection Engine verifies if the updated object belongs to Active Selections. 4 AXEPTool P2P Active Selection Engine downloads the new version of the object and if its eligible as a « loadable » object it is loaded in the AXDB moving it from the AXINDB.
Post-conditions	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.
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

9.2.13 Automatic loading new AXMEDIS Objects with the AXEPTool

UCId	UC_9.2.13
Use case	Automatic loading new AXMEDIS Objects with the AXEPTool
Description	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.
Actors	None
Assumptions	One or more Active Selections have already been performed.
Steps	1 Publication and Monitoring Objects is informed of the new publication. 2 AXEPTool P2P Active Selection Engine is alerted by Publication and Monitoring Objects. 3 AXEPTool P2P Active Selection Engine verifies if the new published objects matches certain features in the Active Selections. 4 AXEPTool P2P Active Selection Engine loads the new object.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	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.
Issues	None

9.2.14 Manual Loading of AXMEDIS Objects with the AXEPTool

UCId	UC_9.2.14
Use case	Manual Loading of AXMEDIS Objects with the AXEPTool
Description	The user wants to move an AXMEDIS object from the AXEPTool IN AXMEDIS Data Base to the AXMEDIS Data Base.
Actors	Content Provider, Aggregator, Integrator
Assumptions	One or more objects are stored in the AXEPTool IN AXMEDIS Data Base

Steps	1 The user, is invoked by the Publication/Loading Rules/Selection Editor, selects For each selected objects in the AXEPTool IN AXMEDIS Data Base. 2 The Loading Tool Engine moves selected objects from the AXEPTool IN AXMEDIS Data Base to the AXMEDIS Data Base: 2.1 The objects are selected by the provided Selections 2.2 The objects are moved
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.

9.2.15 Creation of a loading rule for the AXEPTool

UCId	UC_9.2.15
Use case	Creation of a loading rule for the AXEPTool
Description	The Loading Tool Engine 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	1a) The user opens the Publication/Loading Rules/Selection Editor of the Loading tool engine 2a) The user fills the data required to build a new loading rule. Or alternatively 1b) The user manually selects an AXMEDIS object in the AXEPTool IN AXMEDIS Data Base, query support can be used for this 2b) The user invokes the function “Build rule by example” in both cases The Loading Tool Engine saves the new rule in the Loading Rules repository.
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

9.2.16 Preview an AXMEDIS object content coming from AXEPTool

UCId	UC_9.2.16
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	1 The user chooses to preview an object. 2 The AXEPTool uses a suitable player for this task.. 3 The object is previewed or an error message should be prompted if not possible.
Post-conditions	None
Variations	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.
Asynchronous actions	The user can cancel, stop, close or replay the preview.
Design suggestions	The preview is performed with the players available.
Issues	None

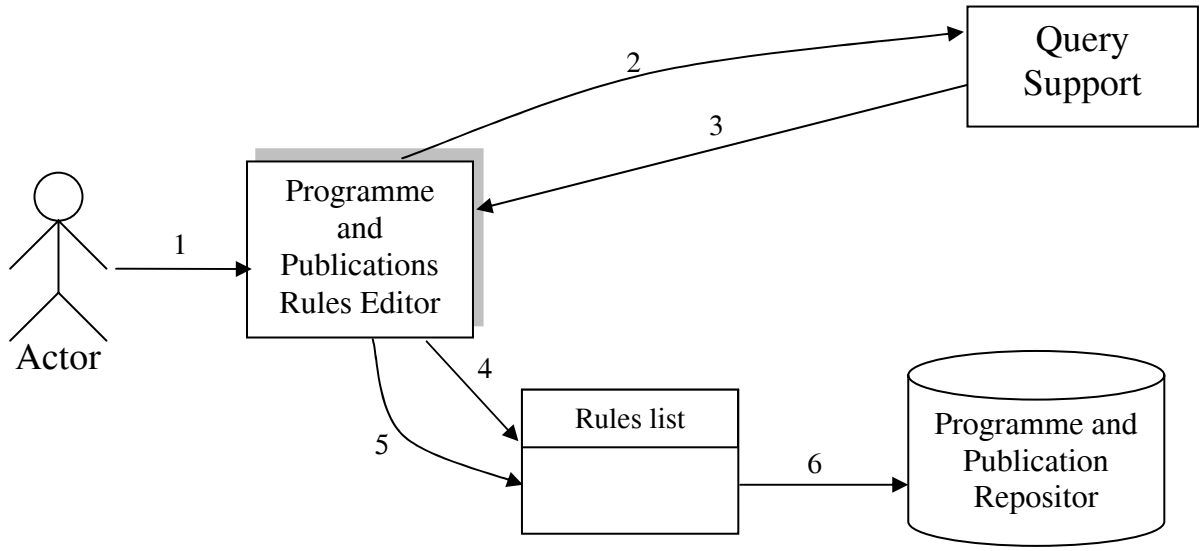
9.2.17 Feedback toward the workflow system

UCId	UC_9.2.17
Use case	Feedback toward the workflow system
Description	The AXEPTool receives by the workflow systems requests of publishing and downloading. At the end of this process the workflow systems has to be informed about the outcome. This use case is intended to be part of an automatic process, so no users are involved in.
Actors	None
Assumptions	The AXEPTool receives a request of publishing or downloading by the workflow system.
Steps	1 If possible, the AXEPTool executes the request. 2 The AXEPTool informs the workflow system about the outcome of the operation.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

10 Programme and Publication Engine Tools

10.1 Programme and Publication Rules Production

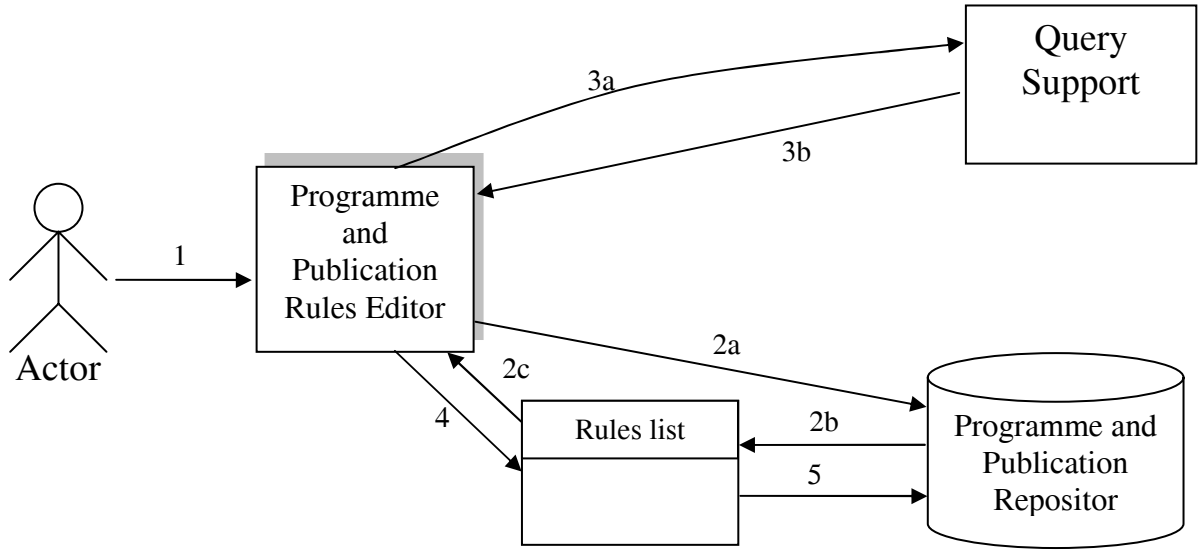
That is how the Rules for Programme and Publication are produced.



UCId	UC_10.1
Use case	Programme and Publication Rules Production
Description	To create/define/edit a programme for certain channel
Actors	A programme producer or programme manager
Assumptions	None
Steps	1 Actor initiates GUI in the Programme and Publication Editor 2 The Actor submits queries to Query Support for a list of AXMEDIS objects 3 Query Support returns a 'selection' (see UCs in section 3.1.2) 4 The Programme and Publication Editor GUI allows the programme producer to select part/all/none of the query results to create a programme (rule list) to state what (objects), where (channel), when (schedule), how (distribution, formatting if necessary), etc. Each of these activities planned may or may not involve formatting or adaptation which is to be checked by comparing the object profile and distribution channel profile. 5 The user specify the distribution channel of this programme 6 The schedule is saved in the Programme and Publication Repository which can be re-used later
Post-conditions	By default the programme is "inactive" at the end of the programme production, until the user activated/published the programme
Variations	None
Asynchronous actions	None
Design suggestions	Requires connection/interface to various other modules including the AXMEDIS Query Support,, etc.
Issues	requires to define a representation for the programme/rule – could be in XML

10.2 Programme and Publication Rules Editing

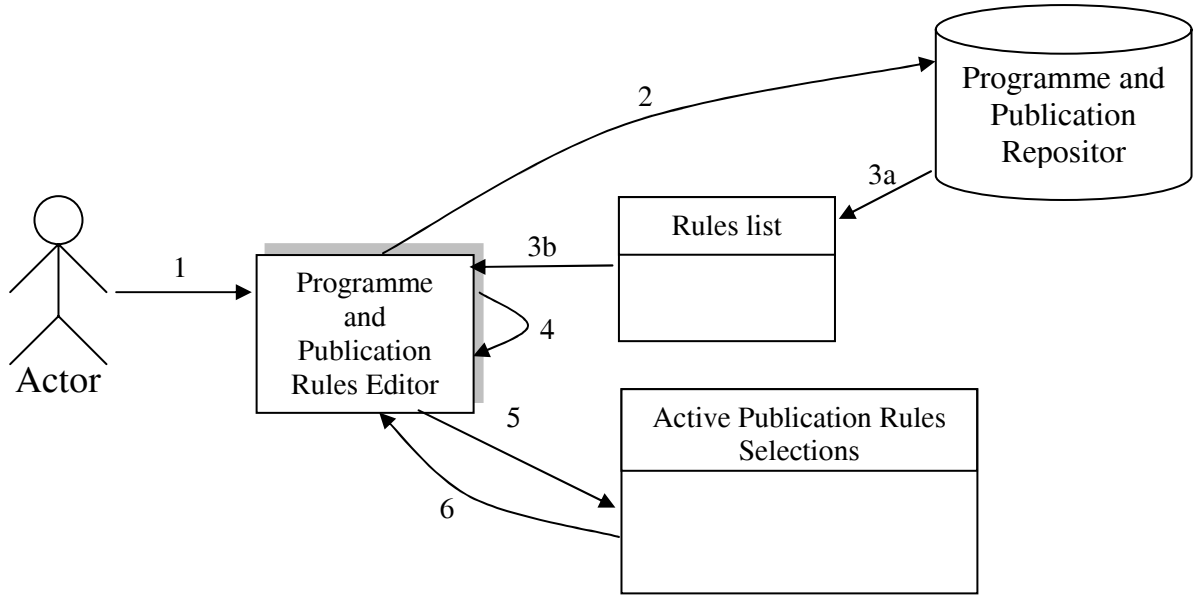
That is how the Rules for Programme and Publication are manipulated.



UCId	UC_10.2
Use case	Programme and Publication Rules Editing
Description	Using the programme and publication tool to edit a programme based on an existing rules
Actors	A programme producer or programme manager
Assumptions	There are one or more predefined/created programme which can be reused, in the collection
Steps	1 Actor initiates GUI in the Programme and Publication Editor 2 (2a) the user browse the existing rules/programme in the collection and (2b) selects schedule for (2c) editing from the Programme and Publication Repository 3 The Actor may (3a) submit queries to Query Support as in UC (the immediate last use case) and (3b) Query Support returns the 'selection' 4 The Programme and Publication Rule Editor GUI allows the programme producer to select part/all/none of the query results to add or edit to a schedule list 5 The new schedule can be saved to the Programme and Publication Repository as in the last use case
Post-conditions	All new rules are also collected and saved in the collection
Variations	None
Asynchronous actions	None
Design suggestions	Requires connection/interface to various other modules including rules collection…
Issues	requires to define a representation for the programme/rule – could be in XML

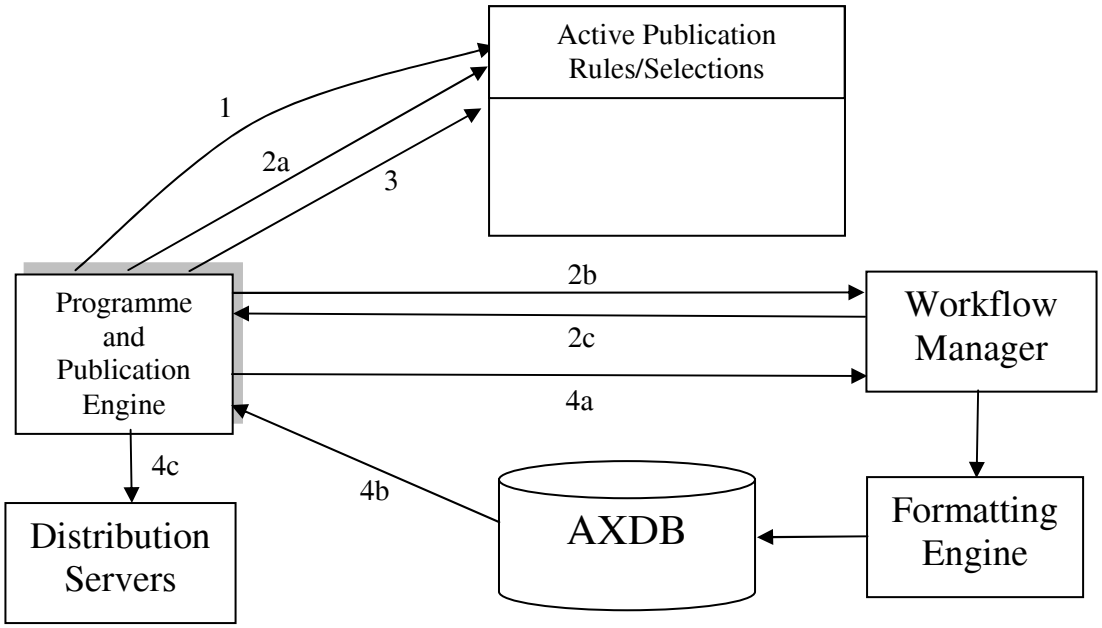
10.3 Activation of Programme and Publication Rules

That is how the Rules for Programme and Publication are activated.



UCId	UC_10.3
Use case	Programme Publication
Description	The user decide the publish ("activate") the programme
Actors	A programme producer or programme manager
Assumptions	A completed programme
Steps	1 The actor uses the Programme and Publication Editor GUI 2 If the programme has not been loaded, the user can select and load the programme, for final checking 3 The programme/schedule is returned from the repository 4 The component check the status and required information of the programme and ask for more input if the programme is incomplete (e.g. unknown publication date or channel) 5 A GUI to allow the user to activate/publish the programme 6 A confirmation on the success of the publication
Post-conditions	None
Variations	None
Asynchronous actions	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
Design suggestions	None
Issues	None

10.4 Launch of Programme and Publication Rules from Workflow



UCId	UC_10.4
Use case	Launch of Programme and Publication Rules from Workflow
Description	This is an active engine which monitors the system clock to ensure that one or more scheduled and published programme is delivered in time for the actual consumption.
Actors	Active Engine
Assumptions	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).
Steps	1 For each rule, 2 if the rule is new (without start time) 2.1 check source and target format 2.2 if profiles mismatch file request to Format using Workflow manager 2.3 returned resulted object or reference 2.4 Get distribution time 2.5 set start time for distribution 3 check publication schedule due for delivery 4 if due, 4.1 request Objects (using the Workflow Manager according to Scenarios v3.6) 4.2 Receive objects (from AXDB according to Scenarioos v3.6) 4.3 send to Distribution Servers 5 Return to 1
Post-conditions	Update the rules collection together with the status of the process
Variations	Content production on demand (request from the AXMEDIS Query Support for Clients via the distributor) also initiates a workflow. In this case the content is produced only for the requesting distribution server.
Asynchronous actions	None
Design suggestions	Requires connection/interface to various other modules including formatting engine, rules collection…

Issues	1. Not sure if this engine is to deliver the object direct to the specified channel or to a distribution server. If to be delivered to the specified channel(s), API for delivery (e.g. ftp?) to every channel is required! In alternative we can suppose to store the objects into the database or into the file system and to send a Acknowledgment to the Workflow or to the Distributor web service to inform where the objects are with a list of them with their references. 2. Presumably the formatting engine can distribute the object to the correct channel, otherwise a more complex interaction between this engine and the formatting engine is required – to wait for the completion of the formatting and to deliver the new object to the distribution server or to deliver direct… 3. How to estimate time requirements for formatting engine and distribution server?

10.5 Trial Pre-activation of Programme and Publication Rules

That is how the Rules for Programme and Publication are pre-activated to simulate, test and be prepared.

UCId	UC_10.5
Use case	Programme Publication Pre-activation
Description	The user decide the publish ("quick trial") or ("full trial") the programme
Actors	A programme producer or programme manager
Assumptions	A completed programme
Steps	1 Steps 1-4 the same as UC_10.3 (Programme Publication) 2 A 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
Post-conditions	None
Variations	None
Asynchronous actions	The user can modify/cancel this action before the completion of the trial schedule distribution.
Design suggestions	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. formatting engine, P&P engine).
Issues	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)

10.6 Launch of Trial Programme and Publication Rules from Workflow

UCId	UC_10.6
Use case	Launch of Trial Programme and Publication Rules from Workflow
Description	This is an active engine which monitors the system clock to ensure that one or more scheduled and published programme is capable for distribution.
Actors	Active Engine

Assumptions	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). If it is a trail run, process it immediately.
Steps	1 For each rule, 2 if the rule is new (without start time) and flagged as a trial (i.e. 1 or 2) 2.1 check source and target format 2.2 if profiles mismatch file request to Format using Workflow manager without requiring formatting 2.3 if quick trial (i.e. flag equal to 1), returned reply from the formatting engine whether formatting is possible otherwise (flag equal to 2) format object and return status from the formatting engine 2.4 Get distribution time 2.5 set start time for distribution 3 if full trial, 3.1 request Objects (using the Workflow Manager according to Scenarios v3.6) 3.2 Receive objects (from AXDB according to Scenarios v3.6) 4 Request distribution server if it is capable of sending this object 5 Return to 1
Post-conditions	Update the rules collection together with the status of the process
Variations	None
Asynchronous actions	None
Design suggestions	Requires connection/interface to various other modules including formatting engine, rules collection…
Issues	1. Formatting engine is not required to format on a quick trial if flagged as a quick trial but is required to reply if formatting is possible or not 2. If full trial, object is retrieved but not sent to the distribution server, required reply form distribution server whether object can be distributed

11 AXMEDIS AXEPTOOLS for Satellite Data Broadcast on B2B

11.1 AXMEDIS B2B Client Application

11.1.1 B2B Client Installation

UCId	UC11.1.1
Use case	B2B Client Installation
Description	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)
Actors	The AXMEDIS professional user
Assumptions	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.
Steps	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 2 The professional user physically installs the DVB-IP adapter according to the installation instructions provided by the manufacturer 3 The professional user connects the DVB-IP adapter to the satellite dish 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 5 The professional user configures the DVB-IP adapter according to instructions 6 The professional user checks that the satellite signal is received correctly 7 The professional user has a special Setup to install the AXMEDIS B2B Client Application 8 The professional user runs the AXMEDIS B2B Client Application Setup 9 The professional user follows the steps of the installation
Post-conditions	The professional user installs other needed Applications useful to treat associated actions with certain AXMEDIS Object.
Variations	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.
Asynchronous actions	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.
Design suggestions	A list of compatible adapters should be prepared. Full installation instructions should be given to the user.
Issues	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.

11.1.2 B2B Client Customization

UCId	UC11.1.2

Use case	B2B Client Customization
Description	The user installs the AXMEDIS Client Application
Actors	The AXMEDIS professional user
Assumptions	The B2B Client Installation has been done successfully.
Steps	1. The professional user configures local and external firewall 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) 3. The professional user modifies (if necessary) some configuration files 4. The professional user disables (if necessary) some complementary module depending on the local configuration (e.g., operating system) 5. The professional user has a stable contact with the Satellite Data Broadcast Provider technical team 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) 7. The professional user keeps up to date the B2B Client Application Component at different layers (drivers, software setup, additional modules) 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
Post-conditions	The professional user will put a special label of quality in the B2B receiving station, certifying the state of art of his installation
Variations	Some scripts could check the correct status of the B2B receiving station and send to a central server detected anomalies.
Asynchronous actions	None.
Design suggestions	None.
Issues	None.

11.1.3 B2B Client Registration

UCId	UC11.1.3
Use case	B2B Client Registration
Description	The professional installer registers the B2B Client Application in order to access the AXMEDIS B2B service
Actors	The AXMEDIS professional user
Assumptions	The professional user has successfully installed the hardware and software AXMEDIS components.
Steps	9. The professional user runs the AXMEDIS Client Application registration procedure 10. The AXMEDIS B2B Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)
Post-conditions	The B2B receiving station is ready to use the AXMEDIS B2B service and receive the AXMEDIS B2B Object.
Variations	The procedure to update the B2B receiving station profile could be automatic and hidden for the system.
Asynchronous actions	None.
Design suggestions	The Server shall manage the B2B receiving station profiles useful to address the content to a part of the B2B users.
Issues	None.

11.2 Enabling a B2B receiving station

UCId	UC11.2
Use case	Enabling a B2B receiving station
Description	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.
Actors	The AXMEDIS Distributor
Assumptions	The AXMEDIS Distributor knows exactly all needed information for registering an authorized B2B station.
Steps	1. The AXMEDIS Distributors accesses to the AXMEDIS User Admin Interface (AXUAI). 2. The AXMEDIS Distributor can manage (add/modify/delete) all receiving station controlled by him
Post-conditions	The AXMEDIS B2B Client, which was just enabled to receive the AXMEDIS B2B carousel, will receive all needed notifications.
Variations	The AXMEDIS Distributor can create one or more groups, and then can associate a receiving station to one or more groups.
Asynchronous actions	None.
Design suggestions	Design a solid environment where the AXMEDIS B2B Client can be simply auto-updated.
Issues	None.

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

UCId	UC11.3
Use case	Download AXMEDIS Object from AXEPTool by using Satellite
Description	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
Actors	The AXEPTool user.
Assumptions	One or more objects are shown as available in the P2P network within a query result list.
Steps	1. The Actor selects one or more objects 2. The Actor chooses the Download Transfer mode (P2P, Satellite Data Broadcast) 3. The Actor (after choosing Satellite Data Broadcast) selects one or more B2B receiving stations (controlled by him) for receiving the previously selected Object 4. The Actor starts the download task in AXEPTool 5. Verification of DRM rules, protections and licensing aspects 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.
Post-conditions	The downloaded object is stored in the Satellite Data Broadcast storage server before sending it.
Variations	None
Asynchronous actions	The Actor can start, suspend, cancel or resume the download task of an object
Design suggestions	Feedback on download status must be implemented.
Issues	None

11.3.1 Pushing an AXMEDIS Object by B2B Carousel

UCId	UC11.3.1
Use case	Pushing AXMEDIS Content by B2B Carousel
Description	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.
Actors	The AXMEDIS Distributor.
Assumptions	The AXMEDIS Distributor is authorized to use the Satellite Data Broadcast like delivery means.
Steps	1. The Actor packages the downloaded content to be compatible with the Satellite Data Broadcast system 2. The Actor selects the group of authorized receiving B2B stations to associate with the AXMEDIS Content 3. The Actor associates the selected Object to a given Programme (the programme is charged of transmitting the Carousel sequence) 4. The Actor schedules the Programme for transmission
Post-conditions	None.
Variations	None.
Asynchronous actions	The Actor can start, suspend, cancel or resume the Programme transmission of the Carousel.
Design suggestions	None.
Issues	None

11.3.2 Updating AXMEDIS Content by B2B Carousel

UCId	UC11.3.2
Use case	Updating AXMEDIS Content by B2B Carousel
Description	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.
Actors	(TBD) The AXMEDIS Synchronizer (?)
Assumptions	The AXMEDIS Synchronizer produces the AXMEDIS Updates to send to distributors. Updates could be produced with daily/weekly basis.
Steps	1. The Actor produces the periodic update 2. The Actor uploads updates packages to the Satellite Data Broadcast storage server (by HTTP, FTP) 3. The Actor packages the uploaded content to be compatible with the Satellite Data Broadcast system 4. The Actor selects the group of receiving B2B stations, dedicated to AXMEDIS Distributor, in order to authorize who has to receive the AXMEDIS Content 5. The Actor associates the selected Object to a given Programme (the Programme is charged of transmitting the Carousel sequence) 6. The Actor schedules the Programme for transmission
Post-conditions	None.
Variations	None.
Asynchronous actions	The Actor can start, suspend, cancel or resume the Programme transmission of the Carousel.
Design suggestions	None.
Issues	None

11.4 Automatic Content Reception via Satellite

UCId	UC11.4
Use case	Automatic Content Reception via Satellite

Description	The AXMEDIS B2B Client Application has detected an AXMEDIS Object addressed to him. He receives automatically the content by push.
Actors	The AXMEDIS B2B Client Application
Assumptions	The AXMEDIS B2B Client Application runs permanently on the AXMEDIS Distributor remote station like a daemon.
Steps	1. The AXMEDIS B2B Client detects an AXMEDIS Object in the Electronic Programme Guide of the Satellite Data Broadcast 2. The AXMEDIS B2B Client checks if it has all rights to listen the incoming transmission of the Object 3. The AXMEDIS B2B Client launches all needed operations in order to receive the AXMEDIS Content
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None.
Design suggestions	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.
Issues	None.

11.5 Content Delivery via Satellite

UCId	UC11.5
Use case	Content Delivery via Satellite
Description	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.
Actors	The AXMEDIS B2B Client Application and other Applications charged of applying actions on the AXMEDIS Object.
Assumptions	None.
Steps	1. The AXMEDIS Client Application receives the last bit of the current transmission and completes the AXMEDIS Object 2. The AXMEDIS Client Application checks the correctness of the received Object (checksum, version numbering) 3. The AXMEDIS Client Application loads the actions to be executed on the Object after reception 4. The AXMEDIS Client Application parses the action list and checks if it is able to treat the content of the action list 5. The AXMEDIS Client Application runs all actions that it can execute directly 6. The AXMEDIS Client Application forwards to other applications (explicitly indicated with the action to execute) all actions that it cannnot execute directly 7. The AXMEDIS Object reaches its final destination
Post-conditions	Typical actions are copy, move, play, apply the AXMEDIS Object.
Variations	Actions on the AXMEDIS Object could be applied before the end of transmission.
Asynchronous actions	The loss of satellite signal in particular weather conditions
Design suggestions	None.
Issues	None.

11.6 Content Protection for Satellite distribution

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

12 AXMEDIS Protection Tools

12.1 Super AXCS

12.1.1 AXMEDIS Registration of AXCSs

UCId	UC12.1.1
Use case	AXMEDIS Registration of AXCSs
Description	An actor wants to register an AXCS in the AXMEDIS system
Actors	Distributor, a company specifically doing that work
Assumptions	The AXCS to be registered is already installed on a machine
Steps	1. The AXCS to be registered is started by the Actor 2. The AXCS contacts the AXMEDIS Registration of AXCSs Web Service providing all the data required for the registration on the Super AXCS. 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 verify also the integrity of the tools, etc. 4. the AXCS store the received data
Post-conditions	The requesting AXCS is registered in the system
Variations	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
Asynchronous actions	None
Design suggestions	None
Issues	None

12.1.2 Tool/device off-line registration

UCId	UC12.1.2
Use case	Tool/device off-line registration
Description	An Actor wants to register a new kind of tool in the AXMEDIS network
Actors	AXMEDIS tool producer (i.e. a software house producing a specified tool to use it in the AXMEDIS system)
Assumptions	The tool is not already registered in the system
Steps	1 Reception of the tool that wants to be registered in the AXMEDIS system 2 Off-line checking and test that tool accomplishes AXMEDIS guidelines 3 If the tool accomplishes AXMEDIS guidelines 3.1 The tool is registered and certified, i.e. tool fingerprint is estimated and other major parameters are extracted and stored into the Super AXCS for verification at each transactions. 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
Post-conditions	• The tool is registered in the AXCS Registration and Certification Database • A new tool type id is generated and bounded to the tool type • The requester receives notification about the registration
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.1.3 AXMEDIS Object ID Generation

12.1.3.1 Generation of unique Object ID

UCId	UC12.1.3.1
Use case	Generation of unique object ID
Description	An actor wants to associate an AXMEDIS Object ID to the newly created object.
Actors	Integrator, Designer
Assumptions	AXMEDIS Editor is opened (or tool using AXOM)
Steps	1 The use case begins when an object creator requests a "New object ID" in the user interface 2 AXOM sends request to PMS Client (DRM support component) to get authorisation for the operation 3 DRM Support component of PMS sends operation request to AXMEDIS OID generator together with information on the object which the identifier is requested for 4 AXMEDIS OID Generator generates the OID with respect to the information received. 5 The OID is returned to the AXOM by DRM Support
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	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.

12.1.4 Global Object List WEB Service

12.1.4.1 Search of AXMEDIS Objects

UCId	UC12.1.4.1
Use case	Search of AXMEDIS Objects
Description	An Actor wants to perform a search in the AXMEDIS database to retrieve a set of AXMEDIS Objects satisfying several conditions
Actors	Users
Assumptions	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.
Steps	1. The Actor contacts the Global Object List WEB Service using an AXMEDIS Tool or a Web browser. In the following statements the programme used to interact with the Global Object List WEB Service is referred as "Client" 2. The Actor compose the query using the client 3. The query is submitted to the Global Object List WEB Service 4. The Global Object List WEB Service contacts the AXCS Database Interface to submit the received query 5. The AXCS Database Interface perform the query over the database and send the retrieved data to the AXCS Database Interface 6. The AXCS Database Interface send the received data to the Global Object List WEB Service 7. The Global Object List WEB Service send the received data to the requesting client

Post-conditions	The Client receives a list of AXMEDIS Objects (according with the sent query) with the pertinent links to retrieve them
Variations	None
Asynchronous actions	None
Design suggestions	Web service
Issues	None

12.1.5 Super AXCS Collector

12.1.5.1 On-line transfer between AXCS and Super AXCS

UCId	UC12.1.5.1
Use case	On-line transfer between AXCS and Super AXCS
Description	Some information managed by AXCS during an AXMEDIS Object usage has to be transferred to Super AXCS. This transfer involves AXCS Synchronizer and Super AXCS Collector
Actors	End user
Assumptions	None
Steps	1 End user uses an AXMEDIS tool to operate on AXMEDIS Protected Objects that are on different distribution channels 2 Protection Manager Support allows only authorized operations on the objects 3 Objects are accessed on different channels and each AXCS stores its Action-Logs 4 Via AXCS synchronizer general information on Objects or information that allow Super AXCS to recover Action-Logs from the different AXCSs are transferred to Super AXCS Collector 5 Super AXCS collects and sores the received information
Post-conditions	Object usage information are transferred from AXCS to Super AXCS
Variations	4a. If connection between AXCS and Super AXCS is not active, AXCS synchronizer store the information to be transferred in a queue called AXCS Synchronizer Queue. Information stored in that queue are transferred to SuperAXCS Collector when the connection returns active.
Asynchronous actions	None
Design suggestions	None
Issues	None

12.1.5.2 Off-line synchronization between AXCS and Super AXCS

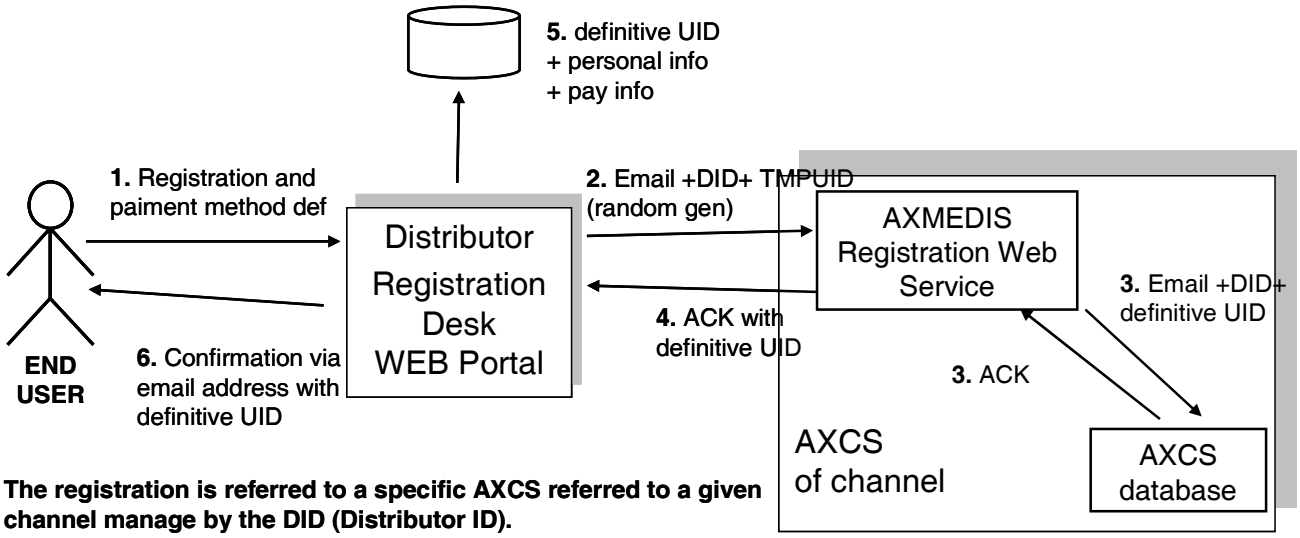
UCId	UC12.1.5.2
Use case	Off-line synchronization between AXCS and Super AXCS
Description	Some information collected by AXCS during an AXMEDIS Object usage has to be transferred to Super AXCS even if the connection between AXCS and SuperAXCS is interrupted. In this case the transfer doesn't occur on-line during the Object usage, but off-line in a second time. This use case describes the line-up between the AXCS database and the SuperAXCS database.
Actors	Super AXCS Collector
Assumptions	None
Steps	1. Super AXCS Collector retrieve the list of all the AXCS registered in the system (performing a query to Active AXCS List Database) 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. 3. The contacted AXCS Synchronizer respond providing the requested data (if present) and empty the AXCS Synchronizer Queue.
Post-conditions	Object usage information are transferred from AXCS to SuperAXCS

Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2 AXMEDIS Certifier and Supervisor

12.2.1 AXMEDIS Registration Service

12.2.1.1 End User registration in a distribution channel



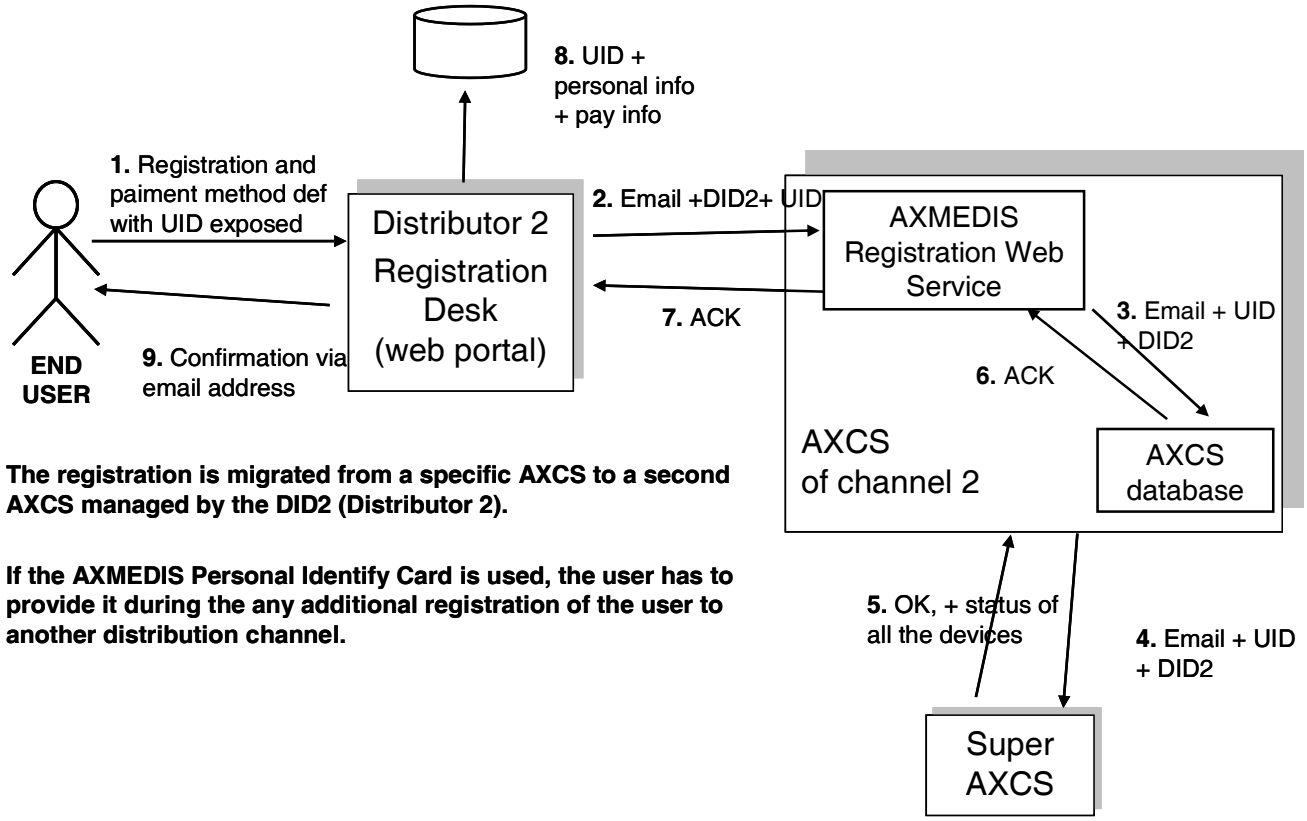
The registration is referred to a specific AXCS referred to a given channel manage by the DID (Distributor ID).

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...

UCId	UC12.2.1.1
Use case	User registration in a distribution channel
Description	An actor wants to register in a channel
Actors	All AXMEDIS Users (we can include: End User, Distributor, Content Provider, Collecting Society and so on)
Assumptions	None
Steps	1 The Actor registers his/her data together with the payment method 2 The Actor's email and DID (Distributor ID) are transferred to the AXMEDIS registration service 3 The AXMEDIS User ID is generated and acknowledged by AXCS 4 An acknowledge with the definitive ID is sent back to the distributor 5 The Actor data are stored into distributor user database 6 A confirmation email is sent back to the Actor
Post-conditions	None
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	None

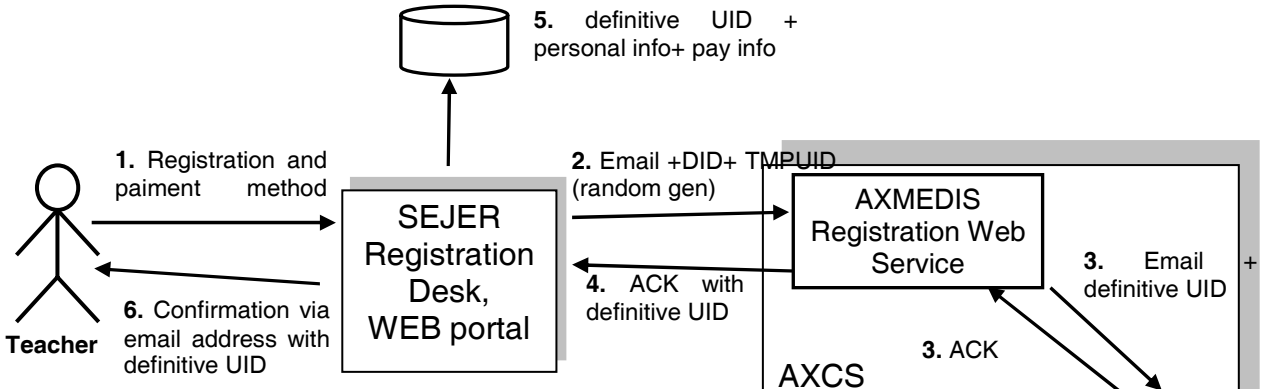
12.2.1.2 End User registration in a different distribution channel



UCId	UC12.2.1.2
Use case	User registration in a second channel
Description	An actor wants to register in a second channel
Actors	End User
Assumptions	None
Steps	1 The Actor registers his/her data together with the payment method. UID is sent to the system 2 The Actor's email, DID and UID are transferred to the AXMEDIS registration service 3 The AXMEDIS User ID in conjunction with new DID and email is sent to AXCS 4 The AXMEDIS User ID in conjunction with new DID and email is sent also to the Super AXCS 5 Super AXCS acknowledge 6 AXCS acknowledge 7 AXMEDIS registration service acknowledges the request 8 User ID is store with other user data in the Distributor user database 9 The Actor receives confirmation of the registration to the distribution channel
Post-conditions	None
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.1.3 Registration of a new Teacher/School or Student



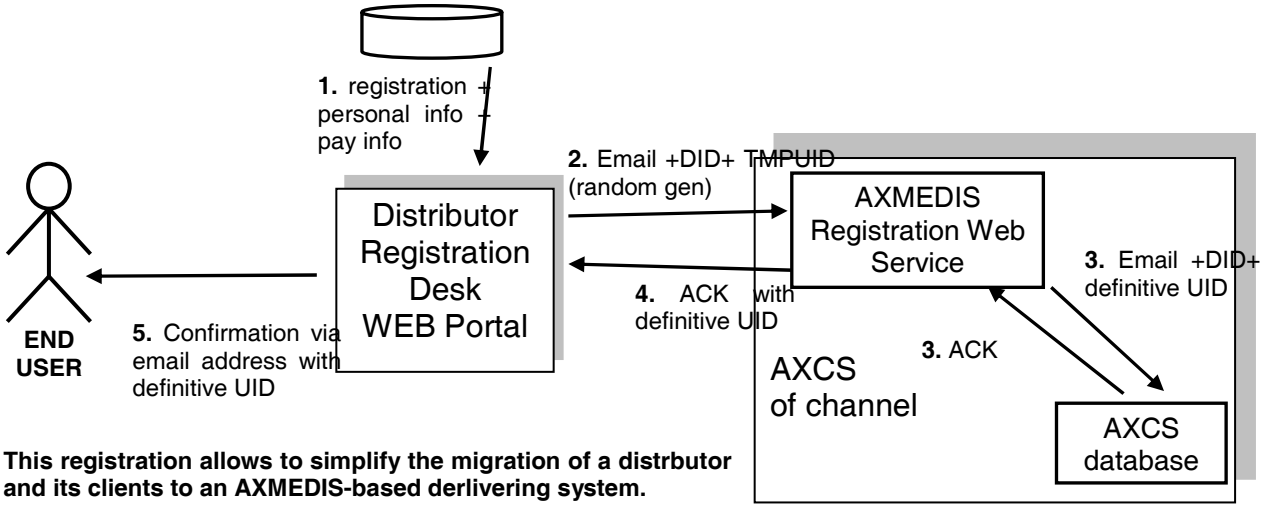
The Teacher/Technician can register himself and all his/her students, my be giving the same email for all the accounts of the students. In that way he can disable those accounts when he/she likes.

Teacher: <UID354135>
Student sdhfgh: <UID134514>
Student afsdhKLFH: <UID675737>
Student dgag: <UID437673>
Student rtywuyert: <UID36773673>

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

UCId	UC12.2.1.3
Use case	Registration of a new Teacher/School or Student
Description	An Actor wants to register in the AXMEDIS network
Actors	Teacher/Technician
Assumptions	None
Steps	1. The Actor registers his/her data together with the payment method 2. The Actor's email and DID (Distributor ID) are transferred to the AXMEDIS registration service 3. The AXMEDIS User ID is generated and acknowledged by AXCS 4. An acknowledge with the definitive ID is sent back to the distributor 5. The Actor data are stored into distributor user database 6. A confirmation email is sent back to the Actor
Post-conditions	• The tool is registered in the AXCS Registration and Certification Database • A new tool type id is generated and bounded to the tool type • The requester receives notification about the registration
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	• The Teacher/Technician can register himself and all his/her students, may be giving the same email for all the accounts of the students. In that way he/she can disable those accounts when he/she likes. • Different UID (or AXMEDIS Personal Identify Cards) will be received by the Teacher, that has to save all the info (emails)

12.2.1.4 Registration of an old User of the Channel on AXMEDIS



This registration allows to simplify the migration of a distrbutor and its clients to an AXMEDIS-based derlivering 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

UCId	UC12.2.1.4
Use case	Registration of an old User of the Channel on AXMEDIS
Description	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
Actors	Distributor
Assumptions	None
Steps	1 The Distributor retrieves the User's information from its database 2 The Distributor sends a registration request to the service filled with data about the user to be registered in the system (see mentioned data in use case "End User registration in a distribution channel") 3 The service checks and validates the data received. Then the service generate a user id, a login, a password for the new user 4 The service sends to the requester the generated data 5 The User receives confirmation of the registration from the Distributor
Post-conditions	• The user is registered in the AXCS Registration and Certification Database • The User receives notification about the registration and user id, login and password
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

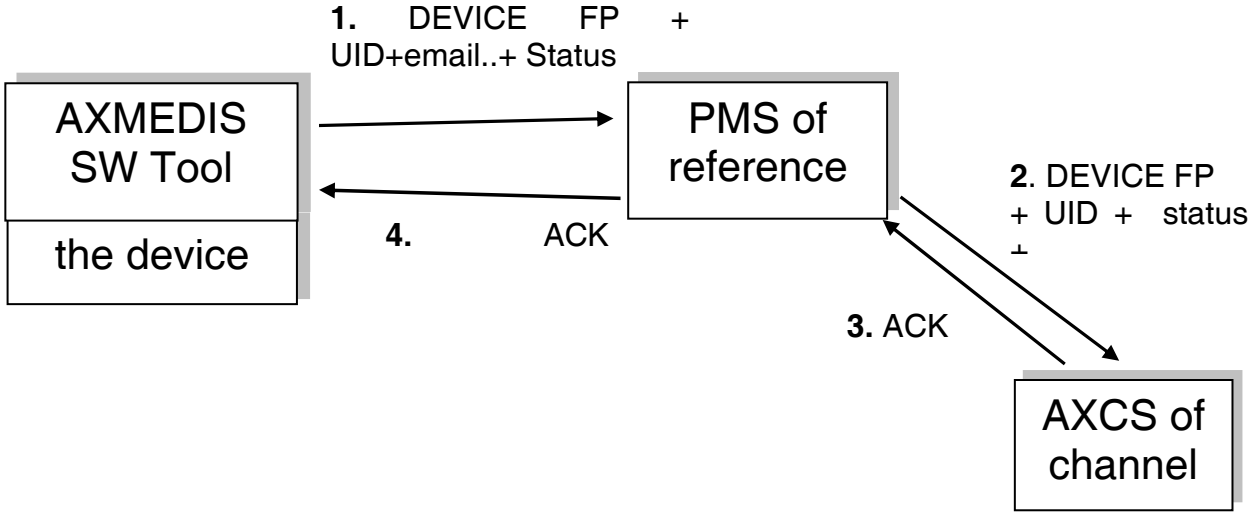
12.2.1.5 User password modification

UCId	UC12.2.1.5
Use case	User password modification
Description	An Actor wants to change a user password
Actors	Content Provider, Distributor
Assumptions	The user is already registered in the system

Steps	1 The Actor sends a password modification request to the service filled with the user id, old password, new password 2 The service checks and validates the data received 3 The service update information related to the user in the database and change the user password as specified 4 The service sends to the requester the confirmation of the password modification
Post-conditions	• The user password is stored in the AXCS Registration and Certification Database • The requester receives notification about the password modification
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.2 AXMEDIS Certification and Verification

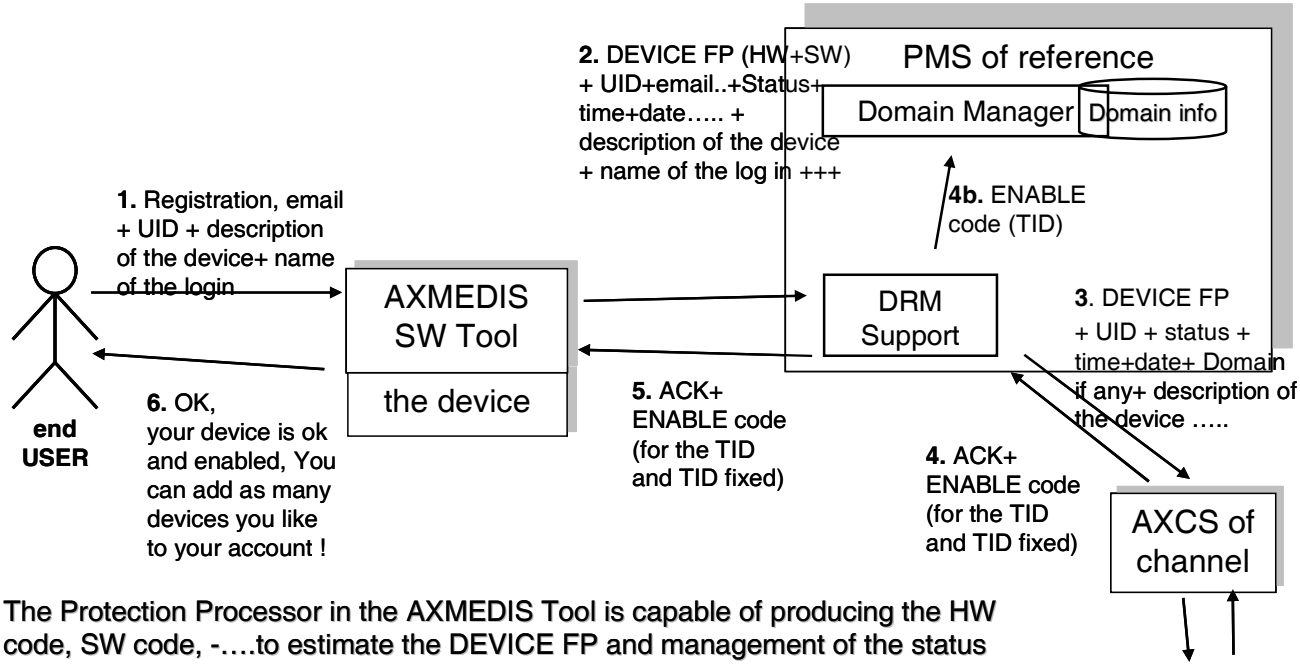
12.2.2.1 Authentication of a Device



UCId	UC12.2.2.1
Use case	Authentication of a Device
Description	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.
Actors	AXMEDIS tool running on a device
Assumptions	The specified tool has been already certified

Steps	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. 2. The PMS sends the received information to the AXCS of channel (eventually adding other) 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 4. The mentioned PMS sends the response to the AXMEDIS Tool: in this way the chain is closed
Post-conditions	None
Variations	If the AXMEDIS Certification and Verification doesn't authenticate the Device this must be deactivate immediately sending him a "Deactivation Signal" and marking him as "blocked"
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.2.2 Certification of AXMEDIS Tool and User



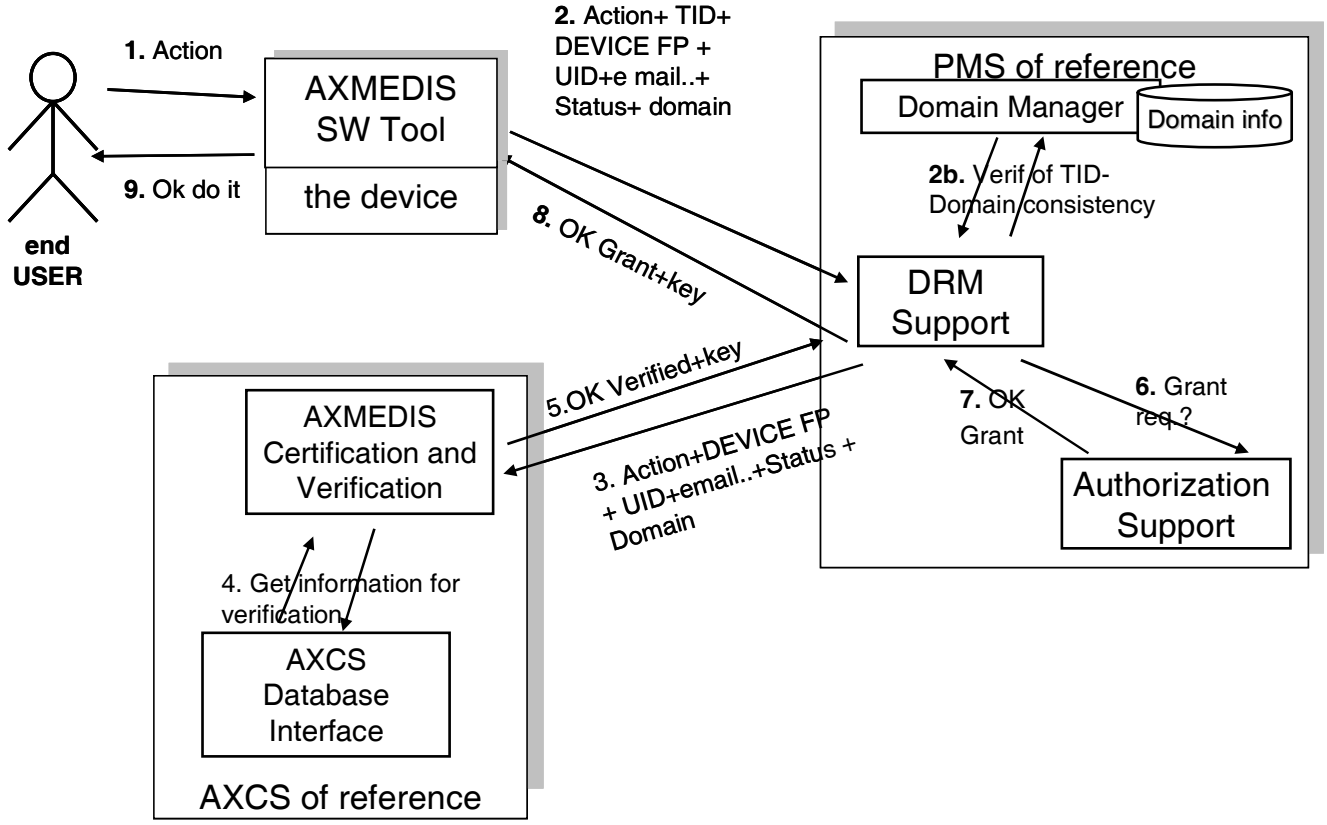
The Protection Processor in the AXMEDIS Tool is capable of producing the HW code, SW code, -….to estimate the DEVICE FP and management of the status depending on time, history of actions, etc. The ENABLE code activate the Tool and will fix for ever the TID of the combination HW+SW+installation.

As a limit case the information maintained on the device by the Protection Processor could be encrypted with a different code for each transaction/verification, this could add dynamism to protection model.

UCId	UC12.2.2.2
Use case	Certification of AXMEDIS tool and user
Description	An Actor wants to certify a specified 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
Actors	AXMEDIS User (can be end user, distributor or whoever uses a tool), AXMEDIS Tool, AXCS Database Interface, PMS
Assumptions	A User wants to use an AXMEDIS tool The specified tool is not already certified: it's the first time it is used

Steps	1 The user opens the tool for its certification 2 AXOM (as a part of the tool) calculates fingerprint or extracts other features to identify the specified tool, the user and the terminal it is installed on 3 AXOM (as a part of the tool) contacts the pertinent PMS sending all the needful information for the registration 4 The mentioned PMS contacts the pertinent AXMEDIS Certification and Verification sending him all the received information 5 Check that it is the first use of the tool by the user. Otherwise see "Verification of AXMEDIS users using AXMEDIS tools during content consumption" use case. 6 If the user and tool are registered (check that the user data and status are correct and that the received Tool FP matches the original one)the AXMEDIS Certification and Verification generates a TID (tool ID) and inserts it together with all the received information in the AXCS Database, using the proper interface, that is, the user is certified to have used the tool for the first time 7 The AXMEDIS Certification and Verification sends to PMS the generated TID 8 The PMS sends to AXOM (as a part of the tool) a certification confirmation message, including the TID 9 AXOM registers that the tool is certified and stores also the received TID 10 If the user or tool are not registered, the PMS and AXOM are sent a message notifying the unsuccessful certification
Post-conditions	If the user and tool are registered, • The tool is certified in the AXMEDIS system • The user is certified to have used the tool for the first time • A new tool id is generated and bounded to the tool • The requester receives notification about the certification
Variations	If the tool is registered but the user is not, the user might be asked to register
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.2.3 Verification of AXMEDIS users using AXMEDIS tools



UCId	UC12.2.2.3
Use case	Verification of AXMEDIS users using AXMEDIS tools
Description	Verification of AXMEDIS users using AXMEDIS tools on a Device during content consumption
Actors	Verify the user data, tool data and tool operation history consistency every time a user wants to use an AXMEDIS tool not for the first time and/or every time the Tool is connected
Assumptions	AXMEDIS User, AXMEDIS Tool, AXCS Database Interface, PMS
Steps	1 A User wants to use an AXMEDIS tool

Post-conditions	2 An AXMEDIS User tries to perform an action on an AXMEDIS Object using an AXMEDIS Tool running on a device 3 The AXMEDIS Tool (AXOM) sends some needful information to PMS, such as: UID, TID, device FP, tool operation history and tool operation history FP, and email 4 The DRM Support inside the PMS of reference contacts the AXCS sending him the received information 5 The Certification and verification (inside AXCS) checks that it is not the first use of the tool by the user. Otherwise see "Certification of AXMEDIS User and Tool" use case 6 Retrieve the tool operation history fingerprint that is stored in the AXCS database (using AXCS Database interface) and match it to the received one 7 If the fingerprints do not match, a new fingerprint is computed, derived from the previous one (stored in the database) and the operation history (sent by the user). The new fingerprint is compared to the fingerprint provided by the user. 8 If the new fingerprint matches the fingerprint provided by the user, the PMS is notified that the user data, tool data and operation history have been verified and is sent the Key 9 The DRM Support (inside the PMS) verifies the DRM using the Authorization Support (inside PMS) 10 The DRM Support responds to AXMEDIS Tool with a Grant signal and the key needed to use the perform the requested action on the AXMEDIS Object The AXMEDIS Tool is now ready to perform the action requested by the AXMEDIS User
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.2.4 Verification 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. In this case the Domain is taken into account.

UCId	UC12.2.2.4
Use case	Verification of AXMEDIS users using AXMEDIS tools on a Device during content consumption inside a domain
Description	The Device/Tool is verified every time an user tries to perform an action
Actors	All AXMEDIS User
Assumptions	A User wants to use an AXMEDIS tool inside a Domain

Steps	1. An AXMEDIS User tries to perform an action on an AXMEDIS Object using an AXMEDIS Tool running on a device 2. The AXMEDIS Tool (AXOM) sends some needful information to PMS, such as: where UID, TID, device FP, tool operation history and tool operation history FP, email and Domain 3. The DRM Support inside the PMS of reference contacts the Domain Manager sending it the received information 4. The Domain Manager performs the verification of TID Domain Consistency 5. The DRM Support receives the response from the Domain Manager and sends the information to AXCS (AXMEDIS Certification and Verification) 6. The AXMEDIS Certification and Verification verifies the received data comparing them with the data retrieved from the AXCS Database (using AXCS Database interface) 7. The AXMEDIS Certification and Verification responds to the PMS sending the Key 8. The DRM Support (inside the PMS) verifies the DRM using the Authorization Support (inside PMS) 9. Insert Action Log into AXCS reporting database 10. The DRM Support responds to AXMEDIS Tool with a Grant signal and the key needed to use the perform the requested action on the AXMEDIS Object 11. The AXMEDIS Tool is now ready to perform the action requested by the AXMEDIS User
Post-conditions	None
Variations	(A) If the AXMEDIS Certification and Verification doesn't authenticate the Device, the device must be deactivated immediately sending it a "Deactivation Signal" and marking it as "blocked" (B) If the DRM Support verifies that the AXMEDIS User is not allowed (not granted) to perform the requested action, the key is not sent to the AXMEDIS Tool and the user remains able to use the system (unless the condition (A) is verified!!!)
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.3 AXMEDIS Supervisor

We have two kinds of blocking manual and automatic.

12.2.3.1 User blocking

UCId	UC12.2.3.1
Use case	User blocking
Description	An Actor wants to block a user in the AXMEDIS network
Actors	Content provider, distributor
Assumptions	The user is already registered in the system
Steps	1 The Actor sends a blocking request to the service filled with the user id 2 The service checks and validates the data received 3 The service update information related to the user in the database to prevent him/her using the AXMEDIS network 4 The service sends to the requester the confirmation of the blocking of the user
Post-conditions	• The user is marked as blocked in the AXCS Registration and Certification Database • The requester receives notification about the blocking
Variations	None
Asynchronous actions	None

Design suggestions	I suggest to allow the blocking from a the WEB service accessed by the Distributor or from a user interface to be done personally by a human.
Issues	None

12.2.3.2 User unblocking

UCId	UC12.2.3.2
Use case	User unblocking
Description	An Actor wants to unblock a user in the AXMEDIS network
Actors	Content provider, distributor
Assumptions	The user is already registered in the system The user is marked as blocked
Steps	1 The Actor sends an unblocking request to the AXMEDIS Supervisor filled with the user id 2 The AXMEDIS Supervisor checks and validates the data received 3 The AXMEDIS Supervisor update information related to the user in the database to unblock him/her. The user may restart using the AXMEDIS network 4 The service sends to the requester the confirmation of the unblocking of the user
Post-conditions	• The user is unmarked as blocked in the AXCS Registration and Certification Database • The requester receives notification about the unblocking
Variations	None
Asynchronous actions	None
Design suggestions	suggest to allow the unblocking from a the WEB service accessed by the Distributor or from a user interface to be done personally by a human.
Issues	None

12.2.3.3 Tool blocking

As stated above we may have manual and automatic blocking of tools. On the other hand, we need additional variations. Blocking a tool can have different "rules":

- Blocking a specific version belonging to one user (e.g. due to manipulations).
- Blocking a specific version (e.g. a new version is available). It is a way to suggest to download a new version. An example of this behaviour is adobe acrobat reader, which informs you about new versions and you decide if you want to download them.
- Blocking a specific version in a mandatory manner only if that version has been cracked. It is a way to force downloading a new version.
- Blocking all versions (e.g. this tool is a general threat to the security)

UCId	UC12.2.3.3
Use case	Tool blocking
Description	An Actor wants to block a tool in the AXMEDIS network
Actors	Content provider, distributor
Assumptions	The tool is already registered in the system
Steps	1 The Actor sends a blocking request to the service filled with the tool id 2 The service checks and validates the data received 3 The service update information related to the tool in the database to prevent it accessing the AXMEDIS network and AXMEDIS Objects 4 The service sends to the requester the confirmation of the blocking of the tool
Post-conditions	• The tool is marked as blocked in the AXCS Registration and Certification Database • The requester receives notification about the blocking

Variations	Blocking all versions of a tool (e.g. this tool is a general threat to the security) Blocking a specific version of a tool if this version has been cracked
Asynchronous actions	None
Design suggestions	None
Issues	Who is responsible of the blocking of the tools? The Super AXCS has to guarantee the security thus if a tool fails is blocked.

12.2.3.4 Tool unblocking

UCId	UC12.2.3.4
Use case	Tool unblocking
Description	A content provider or a distributor wants to unblock a tool in the AXMEDIS network
Actors	Content provider, distributor
Assumptions	The tool is already registered in the system The tool is marked as blocked
Steps	1 An content provider or distributor sends a unblocking request to the service filled with the tool id 2 The service checks and validates the data received 3 The service update information related to the tool in the database to unblock it. The tool may restart accessing the AXMEDIS network 4 The service sends to the requester the confirmation of the unblocking of the tool
Post-conditions	• The tool is unmarked as blocked in the AXCS Registration and Certification Database • The requester receives notification about the unblocking
Variations	None
Asynchronous actions	None
Design suggestions	It is probable that this feature is not needed if a tool has been blocked for some reason and the reason is real there is no reason to unblock, a new version of the tool is needed. That version cannot be used anymore.
Issues	Who is responsible of the unblocking of the tools? The Super AXCS has to guarantee the security thus if a tool fails is blocked.

12.2.3.5 AXMEDIS Protection Information delivery

UCId	UC12.2.3.5
Use case	AXMEDIS Protection Information delivery
Description	AXMEDIS protection Information delivery for supporting protection of AXMEDIS objects
Actors	AXMEDIS User
Assumptions	The Actor has the object
Steps	1 The Actor wants to perform an action over a protected AXMEDIS Object 2 the AXMEDIS Certification and Verification verifies the AXMEDIS Client credentials via PMS 3 PMS calls license manager in order to check if client has the appropriate licenses 4 the AXMEDIS Certifier and Supervisor records the AXMEDIS Client operation in the AXCS Accounting Database 5 the AXMEDIS Supervisor sends the key to the PMS 6 The PMS sends the key to the requesting Actor
Post-conditions	• The Actor has a valid Information for content fruition

Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	Please note that into the protection information is contained. Key, algorithms/rules, etc., for descrambling, or other information, for the protection processor, etc.

12.2.3.6 Storage of protection information of an AXMEDIS Object to the AXCS

UCId	UC12.2.3.6
Use case	Association of protection information to an AXMEDIS Object
Description	Protection information are stored and associated to a given AXMEDIS Object
Actors	None
Assumptions	An AXMEDIS User has protected an AXMEDIS object and generated the related protection information
Steps	1 AXMEDIS Supervisor receives a store request containing the AXOID of the protected object and the related protection information 2 AXMEDIS Supervisor verifies the request validity 3 The protection information are stored in the AXCS database and linked to the given AXOID.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.3.7 Requesting of protection information of an AXMEDIS Object

UCId	UC12.2.3.7
Use case	Get protection information of an AXMEDIS Object
Description	Protection information of an AXMEDIS Object is retrieved
Actors	None
Assumptions	None
Steps	1 Protection information regarding an AXMEDIS object has been requested to the AXMEDIS Database 2 PMS checks that operation is permitted 3 Protection information is returned
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.4 AXMEDIS Reporting Web Service

12.2.4.1 Object usage reporting

UCId	UC12.2.4.1
Use case	Object usage reporting
Description	An Actor wants a report on object usage
Actors	Content provider, distributor, collecting society
Assumptions	None

Steps	1 The Actor sends a report request to the service 2 The service checks and validates the data received 3 The service collects and sends back to the Actor the information related to the object usage
Post-conditions	The Actor has the report
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	The Actor uses Accounting Manager and reporting tool to communicate with AXMEDIS Reporting Web Service

12.2.5 Accounting Manager and Reporting Tool

12.2.5.1 List of all operations performed on an object

UCId	UC12.2.5.1
Use case	List of all operations performed on an object
Description	An Actor wants to know all operations performed on an object
Actors	Content provider, distributor, collecting society
Assumptions	None
Steps	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 2 The service checks and validates the data received 3 The service collects and sends back information related to the object in the database
Post-conditions	The Actor has the list
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.2 List of all operations performed by a user

UCId	UC12.2.5.2
Use case	List of all operations performed by a user
Description	An Actor wants to know all operations performed by a user
Actors	Distributor
Assumptions	None
Steps	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 2 The service checks and validates the data received 3 The service collects and sends back Action-Logs related to the user
Post-conditions	The Distributor has the list
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.3 Usage report about an object

UCId	UC12.2.5.3
Use case	Usage report about an object
Description	An Actor wants to know usage statistic about an object
Actors	Distributor, Collecting society, content owner, content provider

Assumptions	None
Steps	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 2 The service checks and validates the data received 3 The service collects and sends back information related to the object usage
Post-conditions	The Actor obtains the statistic
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.4 Usage report about a distributor

UCId	UC12.2.5.4
Use case	Usage report about a distributor
Description	An Actor wants to know usage statistic about a distributor
Actors	Distributor, Collecting society, content owner, content provider
Assumptions	None
Steps	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 2 The service checks and validates the data received 3 The service collects and sends back information related to the distributor
Post-conditions	The Actor obtains the statistic
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.5 Usage report about a provider

UCId	UC12.2.5.5
Use case	Usage report about a provider
Description	An Actor wants to know usage statistic about a provider
Actors	Distributor, Collecting society, content owner, content provider
Assumptions	None
Steps	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 2 The service checks and validates the data received 3 The service collects and sends back information related to the provider
Post-conditions	The Actor obtains the statistic
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.6 List objects for which an administrative account can be requested

UCId	UC12.2.5.6
Use case	List objects which an administrative account can be requested
Description	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)

Actors	Content creators, distributors, collecting society, content providers
Assumptions	None
Steps	1 An Actor requests the list of objects for which an administrative account can be requested 2 The reporting tool searches in the corresponding database all the AXMEDIS objects for which the actor is eligible to ask a report. 3 The reporting tool lists all the results.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

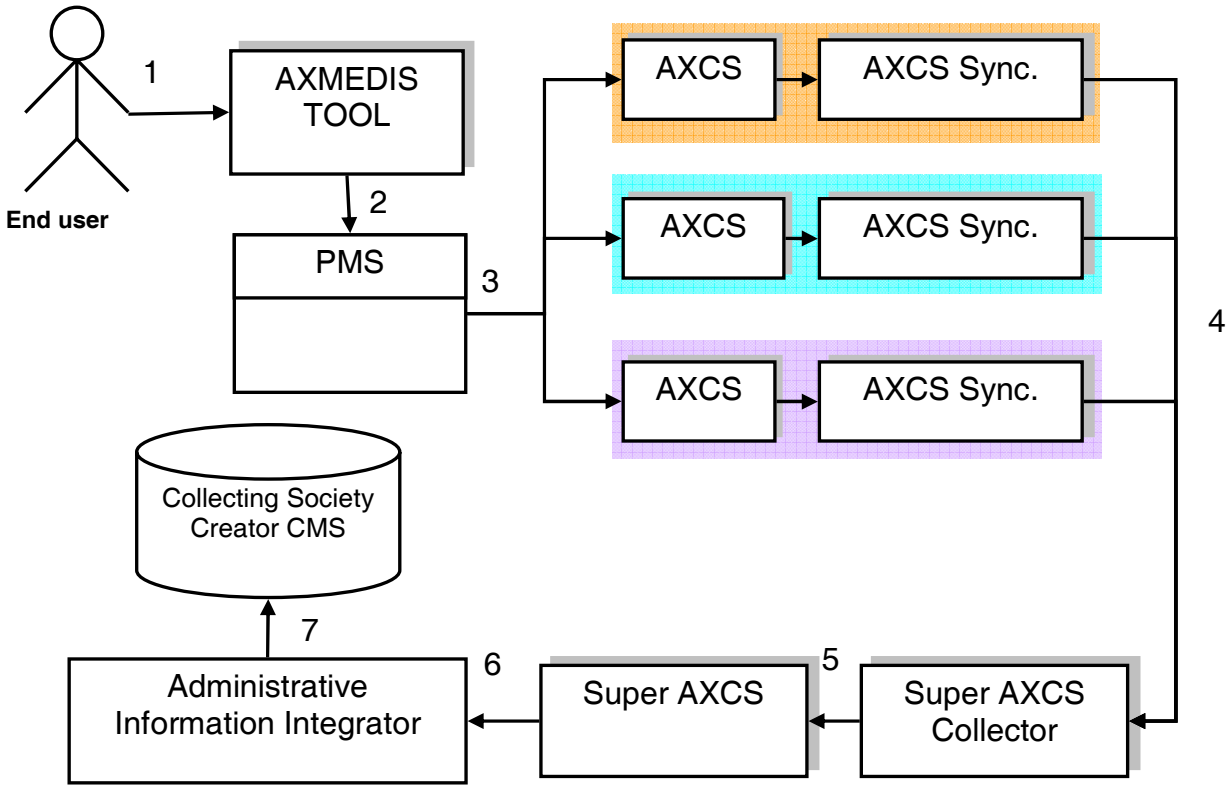
12.2.5.7 Listing AXMEDIS clients of a distributor/channel

UCId	UC12.2.5.7
Use case	Listing all AXMEDIS clients
Description	An Actor wants to consult the AXMEDIS clients list that has been connected to the distributor or on a channel.
Actors	Distributors.
Assumptions	None
Steps	1 The Actor asks for the list of AXMEDIS clients that have been connected to a distributor of a channel 2 The reporting tool searches in the corresponding database all the AXMEDIS clients satisfying point 1. 3 The reporting tool lists all the results.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.2.5.8 Listing distributors

UCId	UC12.2.5.8
Use case	Listing all distributors of AXMEDIS objects, those that have redistributed its objects.
Description	An Actor wants to consult the distributors of AXMEDIS objects.
Actors	Content creators, Distributors, End users, Content Providers.
Assumptions	None
Steps	1 An actor asks for a list of distributors of AXMEDIS objects 2 The reporting tool searches in the corresponding database all the distributors of AXMEDIS objects 3 The reporting tool lists all the results.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

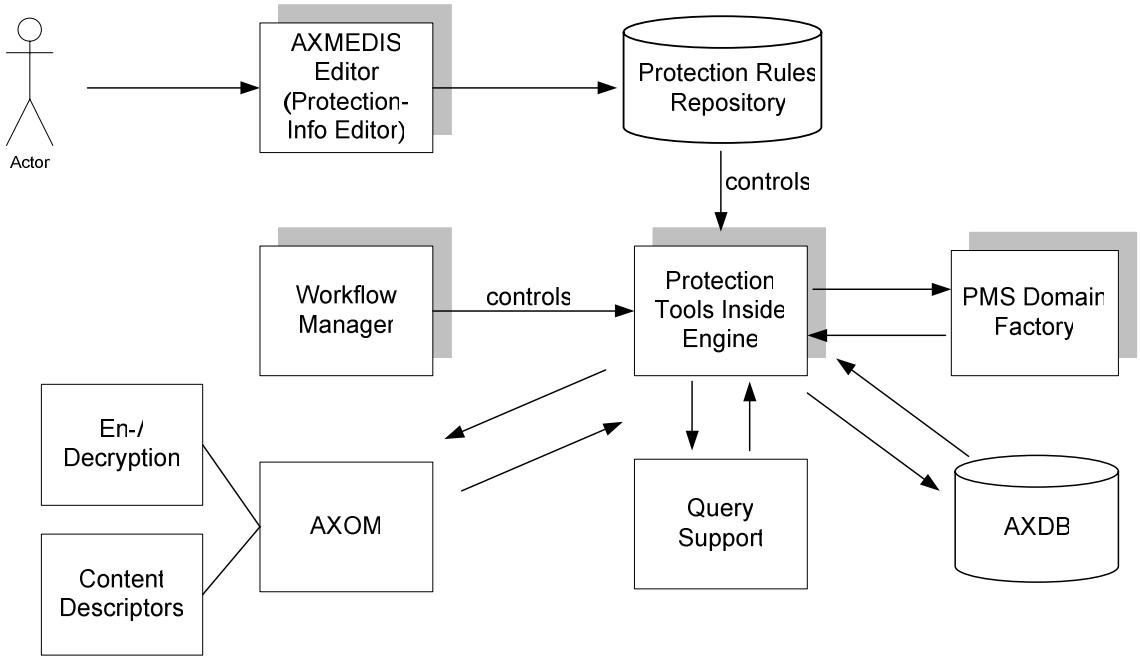
12.2.6 AXCS Synchronizer



1. End user uses an AXMEDIS tool to operate on an AXMEDIS Protected Objects that are on different distribution channels
2. Protection Manager Support allow only authorized operations on the object
3. Objects are accessed on different channels and each AXCS stores its Action-Logs
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
5. SuperAXCS collects information
6. Administrative reports are created
7. Administrative Information Integrator transfer Action-Logs on CMS

12.3 Protection Tool Engine

Protection tool engine use cases regarding DRM support are defined in section 12.5 Protection Manager Support / Server.



12.3.1 Content protection

UCId	UC12.3.1
Use case	Content protection
Description	An AXMEDIS object is protected. The Protection tools engine allows the protection of a large set of objects. Here, protection includes adaptation of PAR and 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.
Actors	Content creator and content distributor.
Assumptions	AXMEDIS object exists and is uniquely identified.

Steps	1 The Protection Tool Engine is initiated to protect an object. This can be done via the WF manager or the Protection Info Editor (with the Protection Rules Repository. 2 Parameters are verified and rule is loaded 3 Content is selected via the Query Support. 4 Content is accessed via the AXOM. 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. 6 PMS Domain factory creates/adaptes the license from the rules or the user input. 7 Verification of PAR or License against given rights 8 PMS Domain factory creates required keys (e.g. for encryption or hash functions). 9 Creation of the protection information 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. 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) 12 Sending the license and the Protection information to the PMS. The PMS forwards the Protection Information to the AXCS.
Post-conditions	AXMEDIS object(s) is (are) encrypted and stored in the AXMEDIS database together with relevant information
Variations	The request can originate from different sources, e.g. • Protection Info Editor (AXMEDIS Editor) • via the AXOM, e.g. from the AXEPTool • via the WF-Manager: This allows the automatic protection of content after its production by the Formatting Tools Engine. Different parameters can be given: • Selection of AXMEDIS object(s) and all protection parameters explicitly • Selection of AXMEDIS object(s) and rules This is the general case including: • Rule based protection of single and multiple objects • Automatic protection during publication via AXEPTool or other tools (e.g. editor)
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.2 Create a new protection rule

UCId	UC12.3.2
Use case	Create a new protection rule
Description	An Actor wants to create a new protection rule
Actors	Content creator and content distributor
Assumptions	None

Steps	1 The Actor creates a Selection of digital resources by making queries to the AXMEDIS Database 2 The Actor rules how these resources have to be protected 3 The Actor stores the created rule into Protection rules Database[FHGIGD: Where are the protection rules stored?]
Post-conditions	None
Variations	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.
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.3 Search and Select a protection rule

UCId	UC12.3.3
Use case	Search and Select a protection rule
Description	An Actor wants to Select a specific protection rule he should be enabled to make some search or browsing.
Actors	Content creator and content distributor
Assumptions	None
Steps	5 The Actor searches in the Protection rules database for a specific protection rule 6 If the Actor finds the rule he can : 6.1 Use it to create a protected AXMEDIS object 6.2 Modify it 6.2.1 Then the Actor store the new rule into the Repository by Protection tool user interface and rule editor 6.2.2 Use the new rule to create a protected AXMEDIS object 7 If the Actor does not find the rule, he can create a new one
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.4 Activating a protection rule

UCId	UC12.3.4
Use case	Activating a protection rule
Description	An Actor wants to activate a protection rule
Actors	Content creator and content distributor
Assumptions	The Protection tool user interface and rule editor can access the Protection Rules Database
Steps	4 The Actor searches in the Protection rules database for a specific protection rule 5 If the Actor does not find the rule 5.1 The Actor can create a new one 6 The Actor selects “Activate Rule” function 4 The rule is put as active rule into the Protection Rules Database
Post-conditions	The rule is added to the set of Active Rule

Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.5 Removing a protection rule

UCId	UC12.3.5
Use case	Removing a protection rule
Description	An Actor wants to remove a protection rule
Actors	Content creator and content distributor
Assumptions	The Protection tool user interface and rule editor can access the Protection Rules Database
Steps	5 The Actor requests the list of Active Rules in the Protection Rules Database 6 The Actor selects the active rule to be disabled 7 The Actor selects "Remove Rule" function 8 The rule is Removed
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.6 Debugging a protection rule

UCId	UC12.3.6
Use case	Debugging/Simulation a protection rule
Description	An Actor wants to debug a protection rule
Actors	Content creator and content distributor
Assumptions	A protection rule is available.
Steps	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
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.3.7 Editing protection rules

UCId	UC12.3.2
Use case	Editing protection rules
Description	A user creates or edits a protection rule.
Actors	Content creator and content distributor..
Assumptions	AXMEDIS object exists and is uniquely identified.

Steps	1 User opens a new or an existing protection rule. 2 User adds, edits, or removes new rules. 3 Protection rule is stored.
Post-conditions	New or modified rules are stored
Variations	None
Asynchronous actions	None
Design suggestions	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.
Issues	None

12.3.8 Printing protection rules

UCId	UCXXX
Use case	Content protection
Description	A user prints the protection rule.
Actors	Content creator and content distributors
Assumptions	eules exists.
Steps	1 The user requests to print the protection rules. 2 Protection rules are printed.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	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 Comp/Form Engine with specific support to work with protection functionalities
The Protection Tool Editor is an instance of the Comp/Form Engine with specific support to work with protection functionalities

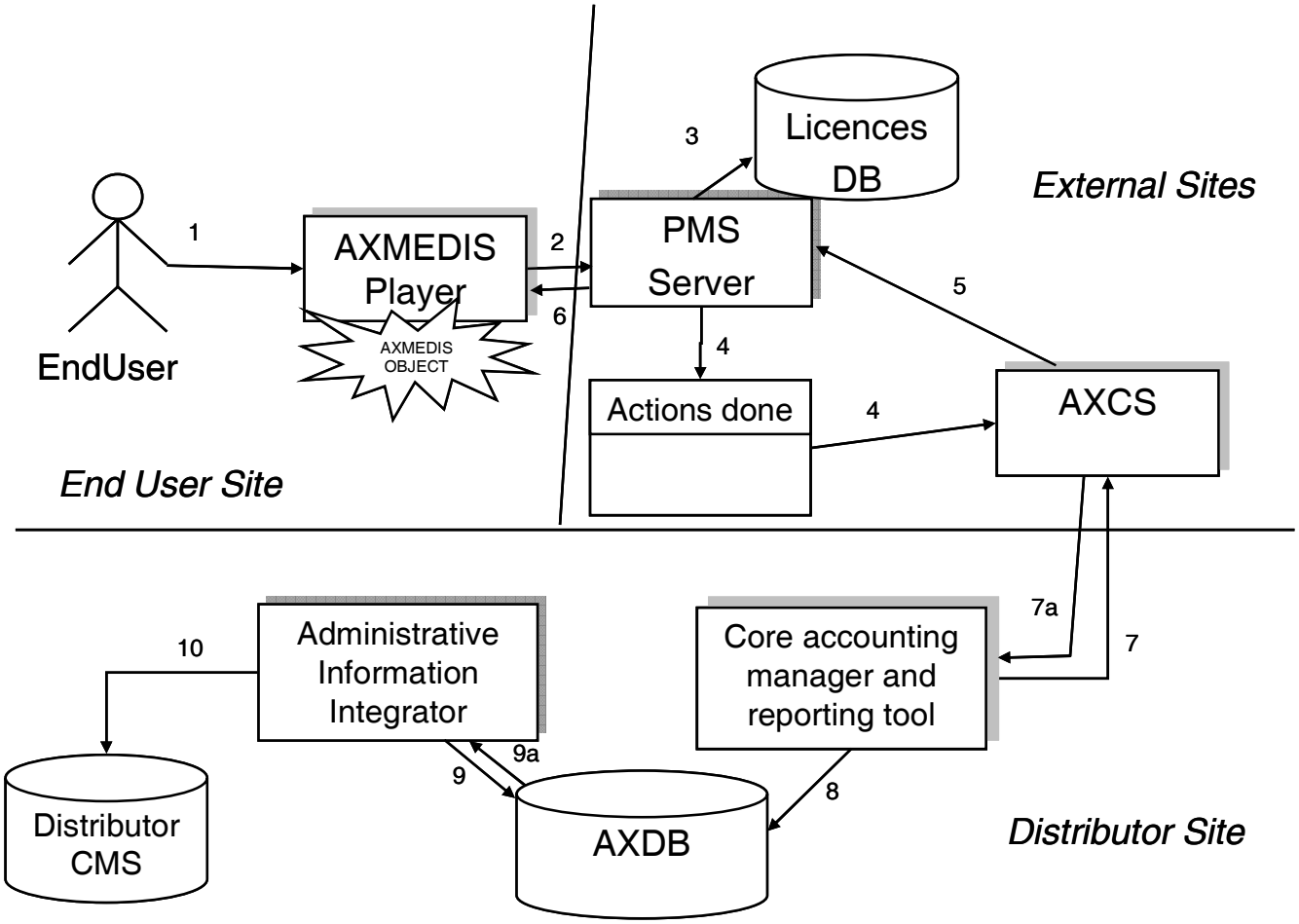
The functionalities that have to be available from the Script and from the Editor are mainly:

- Verification of PAR, internal and external against the License
- Generation of the License from PAR (internal and or external)
- Verification of any PAR or License against some RIGHTS written in clear such as: the play on the AXOID 34 in July 2005 for 5 times, the print of AXOID 56 in Spain in May 2006 at least one, etc.
- Addition/remove of rights from a given License/PAR
- Adaptation of a license/PAR removing some rights, scaling down…
- Production of the Protection Information
- Writing the Protection Information sequence of commands to protect the object (see IPMP standard of MPEG21, plus start-end segment, etc…
- Save the Protection Information sequence to the database

- Save the License into the database
- Send the License to the PMS
- Send the Protection Information to the AXCS via the PMS
- Load from the database of a License Model via the License Model ID
- Code from 0 the rights and other fields that compose a license
- Exploitation of libraries for encryption/decrypt, scramble/descramble, compress/uncompress, etc., as native functionalities into the script and for defining the Protection Information sequence of commands
- Exploitation of lib and DRM support for key generation for Enc/dec Algorithms, couples of keys, etc., different size, etc…

12.4 Administrative Information Integrator

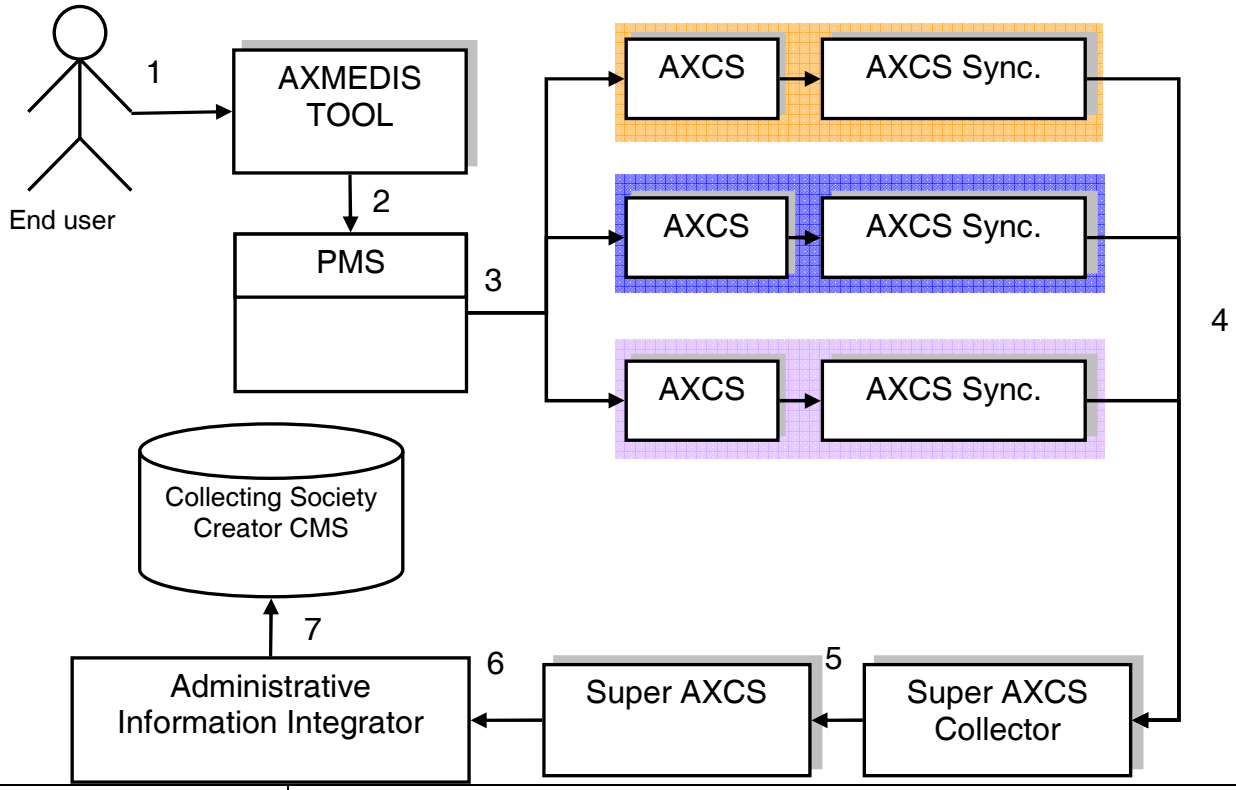
12.4.1 Integrating Distributor administrative information of the basis of End User actions



UCId	UC12.4.1
Use case	Integrating Distributor administrative information of the basis of End User actions
Description	In response to an user action, the administrative information are transferred to the CMS of distributor
Actors	End user
Assumptions	None
Steps	1. End User requests to perform an action on an AXMEDIS Protected Object 2. AXMEDIS Player asks PMS to perform an Action (assuming client has been already certified) 3. PMS checks in the LicenceDB if the Action is allowed (assuming OK) 4. PMS sends AXCS the action performed 5. AXCS gives back the key to access the content (if necessary) 6. PMS gives the grant to access the content and possibly the key to the AXMEDIS Player 7. Accounting & Rep. Tool retrieves from AXCS the actions performed by all the End Users on objects distributed by the distributor 8. A&R Tool stores the transactions in the AXDB 9. Adm. Integrator gets transactions performed from the DB 10. Administrative information are mapped into the Distributor CMS

Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

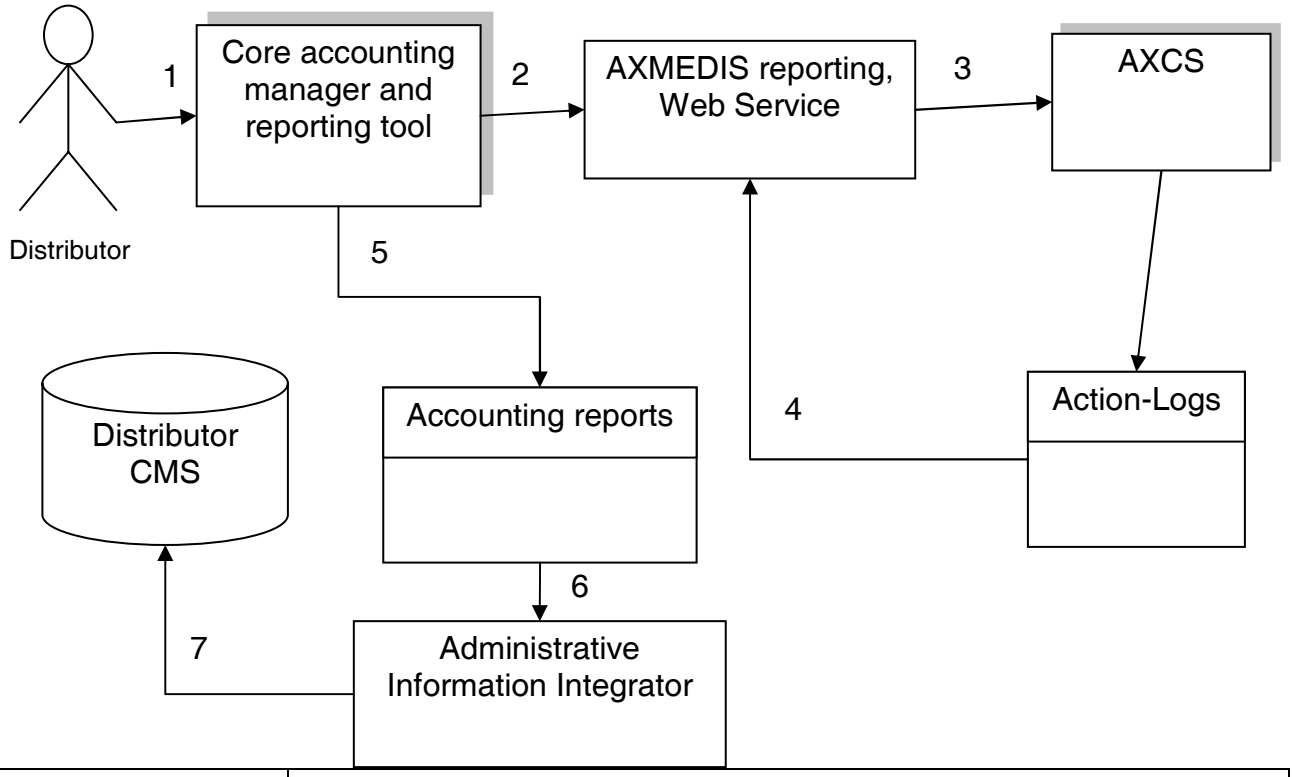
12.4.2 Integrating Collecting Society administrative information of the basis of End User actions



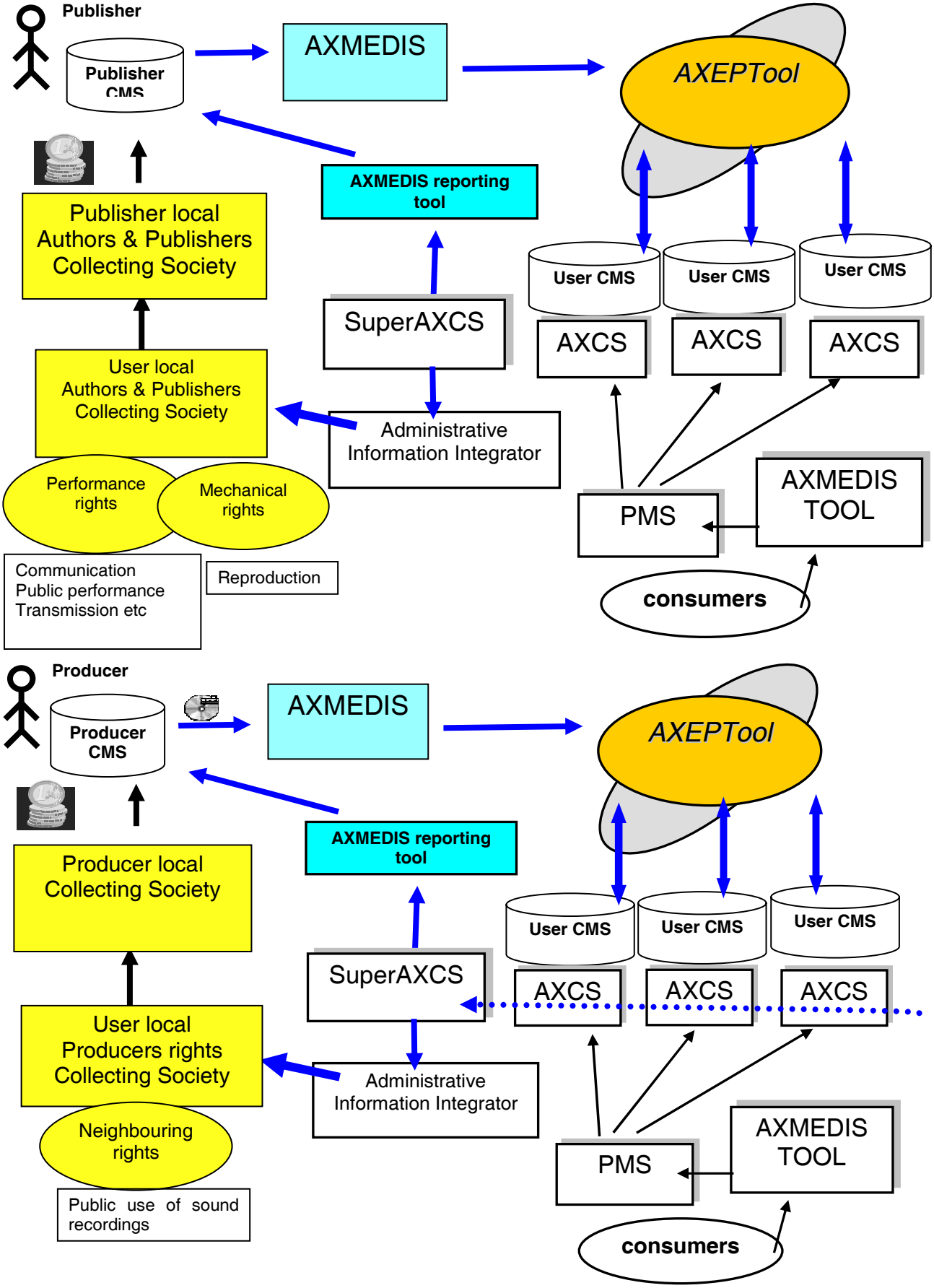
UCId	UC12.4.2
Use case	Integrating Collecting Society administrative information of the basis of End User actions
Description	In response to an user action, the administrative information are transferred to the CMS of collecting society
Actors	End user
Assumptions	None
Steps	1 End user uses an AXMEDIS tool to operate on an AXMEDIS Protected Objects that are on different distribution channels 2 Protection Manager Support allow only authorized operations on the object 3 Objects are accessed on different channels and each AXCS stores its Action-Logs 4 Via the AXCS sync general information on objects or information that allow Super AXCS to recover Action-Logs from the different AXCSs are transferred to the Super AXCS Collector 5 Super AXCS collects information 6 Administrative reports are created 7 Administrative Information Integrator transfer Action-Logs on CMS
Post-conditions	None
Variations	None
Asynchronous actions	None

Design suggestions	None
Issues	None

12.4.3 Distributor asks for administrative information



UCId	UC12.4.3
Use case	Distributor asks for administrative information
Description	In response to the distributor request, the administrative information are transferred to the CMS of distributor
Actors	Distributor
Assumptions	None
Steps	1 A Distributor wants to recover information on actions performed on the objects he has rights. 2 Core accounting manager and reporting tool query the reporting webservice for obtaining the Action-Logs 3 AXMEDIS Statistic or reporting tools query AXCS 4 AXCS extracts the required Action-Logs and communicate them to the reporting tool 5 Accounting report is generated. 6 Accounting report is passed to the Administrative Information Integrator 7 Data are loaded in the Distributor CMS
Post-conditions	None
Variations	None


Publisher
Publisher CMS
AXMEDIS
AXEPTool
AXMEDIS reporting tool
Publisher local Authors & Publishers Collecting Society
User CMS
User CMS
User CMS
SuperAXCS
AXCS
AXCS
AXCS
User local Authors & Publishers Collecting Society
Administrative Information Integrator
Performance rights
Mechanical rights
Communication Public performance Transmission etc
Reproduction
PMS
AXMEDIS TOOL
consumers
Producer
Producer CMS
AXMEDIS
AXEPTool
AXMEDIS reporting tool
Producer local Collecting Society
User CMS
User CMS
User CMS
SuperAXCS
AXCS
AXCS
AXCS
User local Producers rights Collecting Society
Administrative Information Integrator
Neighbouring rights
Public use of sound recordings
PMS
AXMEDIS TOOL
consumers

12.4.4 Administrative information retrieval for distributors

UCId	UC12.4.4
Use case	Administrative information retrieval for distributors
Description	A content distributor requests administrative information about its own AXMEDIS objects
Actors	Content distributor
Assumptions	Content provider is registered and has distributed his/her AXMEDIS objects, users have used objects
Steps	1 A content distributor requests administrative information about its own AXMEDIS objects. 2 the administrative information integrator asks the AXCS to verify the content provider 3 the AXCS certificates the content provider 4 the administrative information integrator collects all Action-Logs related to the distributor objects 5 the administrative information integrator records all collected data into the distributor CMS
Post-conditions	Administrative information is inside the content provider database
Variations	3a. content provider is not registered: 3a.1.the AXCS doesn't validate the content provider 3a.2.the AXCS blocks the operation in progress
Asynchronous actions	None
Design suggestions	None
Issues	None

12.4.5 Administrative information retrieval for collecting societies

UCId	UC12.4.5
Use case	Administrative information retrieval for collecting
Description	A collecting society requests administrative information about the objects under its territorial responsibility
Actors	Collecting society
Assumptions	Collecting society is registered and has to collect rights for objects distributed by different distributors on different channels
Steps	1 A collecting society requests administrative information about AXMEDIS objects under its authority. 2 The administrative information integrator asks the AXCS to verify the collecting society 3 the AXCS certificates the collecting society 4 The SuperAXCS collects informations 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 5 the administrative information integrator collects from the SuperAXCS all Action-Logs related to the objects 6 the administrative information integrator records all collected data into the collecting society CMS
Post-conditions	Administrative information is inside the content provider database
Variations	4a. content provider is not registered: 4a.1.the AXCS doesn't validate the content provider 4a.2.the AXCS blocks the operation in progress
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5 Protection Manager Support/Server general

12.5.1 Protection Manager Support / Server

12.5.1.1 Consumption of a protected and governed AXMEDIS object in a connected environment

UCId	UC12.5.1.1
Use case	Consumption of a protected and governed AXMEDIS object in a connected environment
Description	A user wants to consume a protected and governed AXMEDIS object
Actors	End-user
Assumptions	User is registered. The user is connected.
Steps	1 A user tries to consume a protected and governed AXMEDIS object. 2 As the AXMEDIS object is governed, the PMS obtains the status information from AXMEDIS Certifier and Supervisor. 3 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) and the status information. The authorisation server obtains the licenses associated to the user from the database of DRM licenses, if necessary, and performs the authorisation. 4 If the result of the authorisation is negative, it returns the reasons why not and the user cannot consume the AXMEDIS object. 5 If the result of the authorisation is positive: 5.1 the PMS checks with AXCS if user has got the keys for decrypting object. 5.2 If not, the PMS obtains the secret information (decryption keys) needed to unprotect the object from the AXCS. This information is delivered to the user over a secure channel.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5.1.2 Consumption of a protected and governed AXMEDIS object in a unconnected environment

UCId	UC12.5.1.2
Use case	Consumption of a protected and governed AXMEDIS object in a unconnected environment
Description	A user wants to consume a protected and governed AXMEDIS object
Actors	End-user
Assumptions	User is registered and has previously downloaded the AXMEDIS object. The user does not have connection.

Steps	1 A user tries to consume a protected and governed AXMEDIS object. 2 As the AXMEDIS object is governed, the PMS obtains the status information from its Content Consumption status manager. 3 The PMS request the authorisation to the License Interpreter. It sends an authorisation request that includes the user identification, the right, the resource, the license(s) and the status information. The license interpreter performs the authorisation. 4 If the result of the authorisation is negative, it returns the reasons why not and the user cannot consume the AXMEDIS object. 5 If the result of the authorisation is positive: 5.1 The PMS obtains the secret information (decryption keys) needed to unprotect the object from the Key Manager.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5.1.3 Protection of an AXMEDIS object

UCId	UC12.5.1.3
Use case	Protection of an AXMEDIS object
Description	A user wants to protect an AXMEDIS object
Actors	AXMEDIS Object creator, Protection tool engine
Assumptions	The user has logged in and his identity has been validated. AXMEDIS Editor is running and an AXMEDIS object is opened at the editor
Steps	1 A user wants to protect an AXMEDIS object. 2 AXMEDIS editor makes use of Encryption support (see use case Encryption) 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) 4 PMS sends the protection information to the AXCS, which stores it in the database (See Storage of security information use case)
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5.1.4 Protection and association of licenses of/to an AXMEDIS object

UCId	UC12.5.1.4
Use case	Protection and association of licenses of/to an AXMEDIS object
Description	A user wants to protect an AXMEDIS object and to associate to it license(s).
Actors	AXMEDIS Object creator, Protection Tool engine
Assumptions	The user has logged in and his identity has been validated. AXMEDIS Editor is running and an AXMEDIS object is opened at the editor

Steps	1 A user wants to protect an AXMEDIS object and to associate to the AXMEDIS object a set of licenses. 2 AXMEDIS editor makes use of Encryption support (see use case Encryption) 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) 4 PMS generates the appropriate rights expressions that contain: 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.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

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

UCId	UC12.5.1.5
Use case	Renewal of IPMP information after the detection of a succeed attack
Description	The protection of an AXMEDIS object has been violated, and a renewal of it is needed
Actors	AXCS, PMS
Assumptions	
Steps	1 A succeed attack over the protection of an AXMEDIS object has been detected by AXCS (or external detection). 2 New key for protecting the object is generated 3 The AXMEDIS object is re-protected with the new key (new algorithm) 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
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5.2 DRM Support

12.5.2.1 License creation for new content

UCId	UC12.5.2.1
Use case	License creation for new content.
Description	An actor requests a license to consume or distribute content.
Actors	Content creator, Distributor, Integrator
Assumptions	AXMEDIS object exists and is uniquely identified

Steps	1 The use case begins when an actor requests a license. 2 The License Generator obtains the relevant parameters to generate the license (principal, right/s, resource and conditions). 3 The License Generator creates the license 4 The License Verifier validates the license 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. 6 The License Generator returns to the actor the license ID or the license in clear-text or both.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	Implement it as a web service interface.
Issues	None

12.5.2.2 License creation for cross-media content

UCId	UC12.5.2.2
Use case	License creation for cross-media content.
Description	An actor requests a license to consume, create or distribute cross-media content.
Actors	Content Creator, Distributor, Integrator
Assumptions	AXMEDIS objects that have to be integrated exist, are uniquely identified and have some licenses associated
Steps	1 The use case begins when an actor wants to create a license for cross-media content from several existing AXMEDIS objects 2 The License Generator obtains from the License Manager all the licenses associated to the original AXMEDIS objects 3 The License Generator derives a new license from the obtained licenses 4 The License Verifier validates the new license 5 The License Verifier verifies that the derived conditions are consistent 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. 7 The License Generator returns to the actor the license ID or the license in clear-text or both.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	Implement it as a web service interface.
Issues	None

12.5.2.3 License migration

UCId	UC12.5.2.3
Use case	Content license migration between user domain devices
Description	A user wants to migrate a license from one device to another.
Actors	End-users.
Assumptions	The license contains the information about the user devices it can be transferred to (if no user devices specified, it is valid for all the devices of the user).
Steps	1 If the user has a license, it is transferred unaltered to the device. 2 If not, as it is stored in the license database, the user only has to identify him in the system.
Post-conditions	None
Variations	None
Asynchronous actions	None

Design suggestions	None
Issues	The migration process does not check if the license is usable in the destination device.

12.5.2.4 User authorisation

UCId	UC12.5.2.4
Use case	User authorisation based on licenses
Description	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.
Actors	Distributors, end-users.
Assumptions	Licenses are expressed in MPEG-21 REL.
Steps	1 The authorisation server receives an authorisation request that includes the user identification, the right, the resource and optionally the license(s) or its(their) identifier(s). 2 The authorisation server obtains the licenses associated to the user from the database of DRM licenses. 3 The authorisation server creates the authorisation request and story. 4 The authorisation server performs the authorisation. 5 If the result of the authorisation is positive the user is authorised. If not, it returns the reasons why not.
Post-conditions	None
Variations	Use of ODRL licenses.
Asynchronous actions	None
Design suggestions	Implement it as a web service interface.
Issues	None

12.5.2.5 Navigation of licensing information

UCId	UC15.5.2.5
Use case	Navigation on the licensing information
Description	A user wants to view graphically the information related to his licenses.
Actors	Content creators, distributors and end-users.
Assumptions	None
Steps	1 A user requests to see graphically the information within his licenses. 2 The list of licenses associated to this user is shown. 3 The user chooses one of his licenses. 4 The license is shown graphically 5 The user navigates on the information of the license.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.5.2.6 Rights Expression Translator

UCId	UC12.5.2.6
Use case	Rights Expression Translator
Description	The system wants to translate a valid license (for instance, a mobile profile) from a REL into another, providing interoperability.
Actors	System

Assumptions	A valid license exists
Steps	1 The use case begins when the system detects that a REL translation is needed. 2 Selection of a supported REL destination from a list 3 Translate the license 4 Check if the destination license is valid or not
Post-conditions	None
Variations	If the destination license is not valid, the translation will not be possible and the system will show a message informing of this.
Asynchronous actions	None
Design suggestions	None
Issues	None

12.6 Encryption/Decryption Support

12.6.1 Encryption

UCId	UC12.6.1
Use case	Encryption of AXMEDIS object
Description	An actor wants to save a protected AXMEDIS object
Actors	Any that can save an AXMEDIS object
Assumptions	AXMEDIS Editor is running and an AXMEDIS object is opened at the editor
Steps	1 The use case begins when an actor wants to protect an AXMEDIS object 2 Check if actor has permission to protect the object 3 If actor has permission, the key for encrypting the object is recovered from its storage (license, etc) 4 The object is encrypted and can be saved in a protected manner
Post-conditions	None
Variations	The key for encrypting the object does not exist and it has to be created by calling AXMEDIS certifier The Protection Engine wants to protect some objects
Asynchronous actions	None
Design suggestions	Library providing symmetric encryption functionalities
Issues	None

12.6.2 Decryption

UCId	UC12.6.2
Use case	Decryption of AXMEDIS object
Description	An actor wants to open a protected AXMEDIS object
Actors	Any that can open an AXMEDIS object
Assumptions	AXMEDIS Editor or AXMEDIS Viewer is running or the actor double clicks on an AXMEDIS object to open it (Windows Systems)
Steps	1 The use case begins when an actor calls the "Open object" button on AXMEDIS Editor or AXMEDIS Viewer 2 Check if actor has permission to open the object 3 If actor has permission, the key for decrypting the object is recovered 4 The object is decrypted and AXMEDIS Editor or Viewer can show it
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	Library providing symmetric decryption functionalities
Issues	None

12.6.3 Encryption of symmetric key

UCId	UC12.6.3

Use case	Encryption of AXMEDIS object symmetric key using public key techniques
Description	A symmetric key for an AXMEDIS object is encrypted with asymmetric encrypting techniques for secure storage
Actors	The creator of the AXMEDIS object whose symmetric key is going to be encrypted
Assumptions	A symmetric key for an AXMEDIS object has been generated A pair of public key – private key have been generated for the creator actor
Steps	1 Symmetric key for AXMEDIS object is encrypted with the public component of the creator's asymmetric key
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	Library providing asymmetric key encryption
Issues	None

12.6.4 Decryption of symmetric key

UCId	UC12.6.4
Use case	Decryption of AXMEDIS object symmetric key using public key techniques
Description	A symmetric key for an AXMEDIS object is decrypted using asymmetric encrypting techniques to allow AXMEDIS object decryption
Actors	The actor that wants to use protected AXMEDIS object
Assumptions	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
Steps	1 Symmetric key for AXMEDIS object is decrypted with the private component of the actor's asymmetric key
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	Library providing asymmetric key encryption
Issues	None

12.6.5 Storage of security information

UCId	UC12.6.5
Use case	Storage of encryption/decryption information for a protected AXMEDIS object
Description	The encryption/decryption information of a given AXMEDIS object is stored
Actors	None
Assumptions	An actor has requested to save a protected AXMEDIS object
Steps	1 Symmetric key is encrypted by means of public key techniques (see Encryption of symmetric key use case) 2 Cryptographic information regarding protected AXMEDIS object has to be stored 3 Symmetric key for encrypting / decrypting AXMEDIS object 4 Algorithm used
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

12.6.6 Retrieval of security information

UCId	UC12.6.6
Use case	Retrieval of encryption/decryption information for a protected AXMEDIS object
Description	The encryption/decryption information of a given AXMEDIS object is retrieved
Actors	None
Assumptions	An actor has requested to open a protected AXMEDIS object
Steps	1 Cryptographic information regarding protected AXMEDIS object has to be retrieved 2 Symmetric key for encrypting / decrypting AXMEDIS object 3 Algorithm used 4 Symmetric key is decrypted by means of public key techniques (see Decryption of symmetric key use case)
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

13 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:

UC3.1.9 Acquisition of AXMEDIS objects
UC3.1.10 Viewing/Using of AXMEDIS objects
UC4.1.2 Load and save AXMEDIS objects
UC 4.2.1 Invoking an internal viewer (/editor)
UC 4.2.2 Managing a digital resource by respecting the DRM in an Internal Viewer/Editor
UC 4.2.3 Closing an internal viewer (/editor)
UC 4.3.1 Invoking an external tool with a digital resource belonging to the AXMEDIS object
UC 4.3.5 Transferring a digital resource to an external tool
UC 12.6.2 Decryption
UC 12.6.4 Decryption of symmetric key
UC 14.2.1 User Software Installation
UC 14.2.2 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).

13.1 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

13.1.1 Content Recording for Playtime Shift

UCId	UC13.1.1
Use case	Content Recording for Playtime Shift
Description	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.
Actors	The Content Consumer (user)

Assumptions	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.
Steps	1 The user select from a content distributor catalogue an AXMEDIS Object to download 2 The client terminal, if license terms for the Object allow this, activate the Backup Function 3 The user specifies the "title" with which the AXMEDIS Content has to be recorded. 4 The user execute the Backup/Record Function 5 At a later time, the Player can be started in playback mode to play a selected recorded "title".
Post-conditions	None.
Variations	None.
Asynchronous actions	The AXMEDIS Content can be deleted during the playback operation, according to the license terms.
Design suggestions	The Player should have a playback function allowing access to stored "titles" without an explicit path access for the user.
Issues	None.

13.1.2 Fast-forward of Content in Internal Players/Viewers

UCId	UC13.1.2
Use case	Fast-forward of Content in Internal Players/Viewers
Description	The User wants to play a digital resource faster for a quick preview or for fast access to a later sequence
Actors	The Content Consumer (user)
Assumptions	• AXMEDIS Player is open • An object is opened within the AXMEDIS Player
Steps	1 The User select the Play command 2 The system activates the proper internal player/viewer. 3 The User select the fast-forward command 4 The activated viewer/player inside the AXMEDIS Player starts skipping frames at appropriate rate to speed-up playback speed. 5 When the User releases the fast-forward command, the viewer/player returns to normal playback mode.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	End User usually can only "play" an AXMEDIS object, so only internal "players/viewers" should be invoked, not editors.

13.1.3 Local adaptation of Content in Internal Players/Viewers

UCId	UC13.1.3
Use case	Local adaptation of Content in Internal Players/Viewers
Description	The User wants to play more digital resources possibly requiring a resource management in real-time
Actors	The Content Consumer (user)

Assumptions	• AXMEDIS Player is open • One or more objects are opened by the user within the AXMEDIS Player
Steps	1 The User select the Play command 2 The system activates the proper internal player/viewer. 3 The User select again the Play command for a second Object 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. 5 When the User stops one of the object playbacks, the viewer/player returns to normal playback mode.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	End User usually can only "play" an AXMEDIS object, so only internal "players/viewers" should be invoked, not editors.

13.1.4 Annotate for personal use

UCId	UC13.1.4
Use case	Annotate for personal use
Description	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. 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.
Actors	The Content Consumer (user)
Assumptions	
Steps	1 The end user starts playing content. 2 The end user exploits the Player reduced editing functions to add some annotation related to the played content. 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. 4 The final user finishes playing and annotating the content. 5 Annotations are stored independently from AXMEDIS Objects containing the content but keeping a reference to them.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	This can be realized by adding a limited part of the editor functionality and allowing to save notes in a different way
Issues	None

13.1.5 Local User Profiles

UCId	UC13.1.5
Use case	Local User Profiles
Description	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.
Actors	The Content Consumer (user)
Assumptions	
Steps	1 The end user configures and saves a user profile. 2 Every time the Player is opened it checks the user profile and adapts itself to the user preferences.
Post-conditions	None
Variations	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.
Asynchronous actions	None
Design suggestions	None
Issues	None

13.1.6 History of the last played contents

UCId	UC13.1.6
Use case	User profiles
Description	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.
Actors	End User
Assumptions	
Steps	1 Every time the user plays content the Player keeps track of it in a local list. 2 The user can know what the last played contents are.
Post-conditions	none
Variations	none
Asynchronous actions	none
Design suggestions	
Issues	

14 AXMEDIS for Distribution via Internet

14.1 Back Office Management

14.1.1 Creating a New Mediaclub

UCId	UC14.1.1
Use case	Creating a new Mediaclub setup
Description	Set up a new mediaclub in the cms

Actors	System Manager (sys mng)
Assumptions	The system is up and running and fully configured; Actors have network access to the management interface (web). All techinical info needed to configure the mediaclub are provided by the Content distributor
Steps	1 (sys mng) log in to the system and add a new project (name and description) 2 (sys mng) configure the mediaclub website publishing targets and publishing modes (static pages, dynamic, etc) 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 4 (sys mng) creat the project media repository witch contains binaries content as images, video stream, audio stream, etc 5 (sys mng) Define feed import rules 6 (sys mng) Define referred publishing rules, if needed 7 (sys mng) configure the project administrator 8 (sys mng) Save configuration
Post-conditions	Test page should be displayed in the website publishing targets and prj mng successfully log in
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.2 Mediaclub Setup

UCId	UC14.1.2
Use case	Mediaclub set up
Description	Define all mediaclub feactures in the cms
Actors	Project Manager (prj mng)
Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	1 (prj mng) log in to the system and load the project settings form (name and description) 2 (prj mng) configure the mediaclub website sections 3 (prj mng) create the projects content types (xsl schema; xsl target and taget layout) 4 (prj mng) create content categories and media categories three
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.3 Mediaclub Accounts and Permission Management

UCId	UC14.1.3
Use case	Mediaclub accounts and permissions
Description	Manage a mediaclub accounts and their permissions
Actors	Project Manager (prj mng)

Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	1 (prj mng) log in to the system and load the project account management form (n 2 (prj mng) Create a new project account defining personal details, user id, password 3 (prj mng) Define account permission (Editor, publish authorizer, project manager …..)
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.4 Mediaclub Project Uploading and publishing contents

UCId	UC14.1.4
Use case	Mediaclub publishing
Description	Upload contents in the cms and publish them in the related mediaclub site
Actors	(editor) actors allowed to put contents in the mediaclub, (publisher) actors allowed to autorize content publishing
Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	1 (editor) log in to the system and loads the select new content action 2 (editor) choose the content type and define a content name 3 (editor) fill all fields required from the defined content type 4 (editor) save content and choose one or more publishing targets 5 (editor) submit content to authorization for publishing 6 (publisher) authorize or reject the publish request
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.5 Mediaclub Project Acquiring AXMEDIS content

UCId	UC14.1.5
Use case	Mediaclub and AXMEDIS content
Description	Set up a new mediaclub in the cms
Actors	Project Manager (prj mng)
Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	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 mediaclub contents
Post-conditions	None
Variations	None

Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.6 Mediaclub Project define payment gateway entry

UCId	UC14.1.6
Use case	Mediaclub payments system setup
Description	Enable the payment gateway to provide payment service to the specific mediaclub
Actors	System Manager (sys mng)
Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	1 (sys mng) log in to the system and go in to payment management section 2 (sys mng) configure a new mediaclub shop in the payment gateway giving (name, description, other details) 3 (sys mng) Define payment methods available for the mediaclub 4 (sys mng) configure the shop administrator 5 (sys mng) Save configuration
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.7 Mediaclub Shop payment Management

UCId	UC14.1.7
Use case	Mediaclub shop payments configuration
Description	Configure a mediaclub shop in the payment gateway
Actors	Shop Manager (shop mng)
Assumptions	The system is up and running and fully configured; actors have network access to the management interface (web)
Steps	1 (shop mng) log in to the system and go in to payment management section 2 (sys mng) configure mediaclub call back URL for success, failure and error transaction 3 (shop mng) Choose payment methods available for the mediaclub 4 (sys mng) upload schema and graphical components needed to build the payments transaction pages that will be shown to the end user
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.1.8 Mediaclub Shop Management refound a transaction

UCId	UC14.1.8
Use case	Mediaclub refound management

Description	refound a payment transaction in a mediaclub shop
Actors	Shop Manager (shop mng)
Assumptions	Customer have provided transaction details and is proven that he hasn't had the digital goods
Steps	1 (shop mng) search the transaction id and or the user id in the transaction list 2 (shop mng) load the transaction details and check if everithing is ok 3 (shop mng) starts transaction refound process
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

14.2 End User Client configuration

14.2.1 User Software Installation

UCId	UC14.2.1
Use case	User Software Installation
Description	The user installs the AXMEDIS Client Application
Actors	The Content Consumer (user)
Assumptions	User has PC or STB based on most common Operating Systems: Windows XP, 2000, ME; MacOS X; Linux. User has an Internet connection (DSL preferably)
Steps	1 The user obtains the AXMEDIS Client Application Setup (e.g., from the Internet, from a CD, ...) 2 The user runs the AXMEDIS Client Application Setup 3 The user follows the steps of the installation 4 Some information entered by the user is used to create his profile, that is securely communicated via Internet to MediaClub installation
Post-conditions	The system shall have entered the next procedural step
Variations	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).
Asynchronous actions	None.
Design suggestions	Different Setups applications could be distributed according to country, bundling with Tiscali DSL set-up packages, commercial promotions, etc. The Setups should contain minimal 'intelligence' and be driven by the client management server, in order to make updates easier. The client media Server shall implement algorithms to detect the user language and provide him useful information on the service.
Issues	None.

14.2.2 User Registration

UCId	UC14.2.2
Use case	User Registration
Description	The user register himself in order to access the MediaClub service
Actors	The Content Consumer (user)

Assumptions	The user has successfully installed the software AXMEDIS components and has an email address
Steps	1 The user runs the MediaClub web service registration procedure 2 The user enters or updates his personal profile, including user e-mail and marketing information, that is securely stored in the MediaClub 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.) 4 The user replys via email to confirm registration 5 The AXMEDIS Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)
Post-conditions	The user is ready to use the MediaClub service and access the published Content catalogue. (Access can be restricted only to some components)
Variations	The user may re-run the procedure to update his profile. In some situations this procedure could be automatic and hidden to the user. 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.
Asynchronous actions	None.
Design suggestions	user profile may be used in order to personalize content presentation, messages, etc.
Issues	None.

14.3 User login

14.3.1 Authentication trough AXMEDIS client

UCId	UC14.3.1
Use case	User Login
Description	The user authenticate itself into the AXMEDIS system, via the AXMEDIS client (player/editor etc.)
Actors	Any kind of user
Assumptions	User has a properly installer AXMEDIS client on his Device User is already registered in the AXMEDIS system
Steps	1. The user launch the AXMEDIS client software 2. The software prompt the User for its login/password/certificate path or whatever is used to authenticate him 3. The User authenticate itself 4. The AXMEDIS Client application starts a Session for this user, granting the rights associated to his profile
Post-conditions	
Variations	Models where the player is certified for a given device only and can run unrestricted on this device ?
Asynchronous actions	None.
Design suggestions	
Issues	None.

14.3.2 Authentication trough an external SSO system

UCId	UC14.3.2
Use case	User Login

Description	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.
Actors	End users
Assumptions	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
Steps	1. The user launch the VLE client software (?) 2. The software prompt the User for its login/password/certificate path or whatever is used to authenticate him into the VLE 3. The User authenticate itself 4. The VLE checks authorisations and grant rights Accordingly. 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. 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.
Post-conditions	
Variations	The agreement between the VLE and content provider may take many forms : 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 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.
Asynchronous actions	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)
Design suggestions	
Issues	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). This disposal should not break any AXMEDIS security rule.

14.4 Catalogue Browsing

14.4.1 Catalogue Listing

UCId	UC14.4.1
Use case	Catalogue Listing
Description	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 uncast or multicast mode depending if content is on demand or live.
Actors	The Content Consumer (user)

Assumptions	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.
Steps	1 The user reaches the AXMEDIS Catalogue List 2 AXMEDIS content is displayed according to various criteria (type, author, content producer, production date) 3 user selects and accesses content by clicking on the reference
Post-conditions	User accesses Content Access or sub-catalogue List
Variations	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.
Asynchronous actions	User must be opted with FAQ, HELP and customer care forms urls
Design suggestions	The user may search catalogue based on key words or free text search Possible previews related to the AXMEDIS Object may be provided
Issues	None.

14.4.2 Catalogue Searching

UCId	UC14.4.2
Use case	Catalogue Searching
Description	The user searches content in the MediaClub or Content on P2P network
Actors	The Content Consumer (user)
Assumptions	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.
Steps	1 The user accesses a form within the AXMEDIS Client plug-in where he can operate keyword or free-text searches 2 Search results of AXMEDIS content is displayed according to various criteria (type, author, content producer, production date) and wether content is available in MediaClub or on the P2P network 3 user selects and accesses content by clicking on the reference
Post-conditions	User accesses Content Access
Variations	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.
Asynchronous actions	none
Design suggestions	none
Issues	None.

14.4.3 Available resources listing

UCId	UC14.4.3
Use case	Available resources listing
Description	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.

Actors	The Content Consumer (user)
Assumptions	The user has an account on the content distribution system ; his client environment is properly configured to list his available content after logged in.
Steps	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
Post-conditions	Once logged in, each time the user reaches the portal, the list of resources available to him is displayed
Variations	
Asynchronous actions	
Design suggestions	
Issues	None.

14.4.4 Content Access

UCId	UC14.4.4
Use case	Catalogue Content Access
Description	The user selects the Catalogue content
Actors	The Content Consumer (user)
Assumptions	The user has selected content from a Catalogue listing
Steps	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
Post-conditions	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
Variations	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.
Asynchronous actions	none
Design suggestions	Catalogue Content page may include editorial text, picture data
Issues	None.

14.4.5 User Page

UCId	UC14.4.5
Use case	User Profiling
Description	
Actors	The Content Consumer (user)
Assumptions	The user has successfully registered in order to access the MediaClub
Steps	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
Post-conditions	On subsequent access to the MediaClub user is prompted with customized pages that are assembled based on content preferences and on marketing profile

Variations	The user may re-run the procedure to update his content preferences.
Asynchronous actions	none
Design suggestions	(TBD)
Issues	Legal disclaimer for privacy

14.5 Catalogue Content Purchase

14.5.1 Content Fetching

UCId	UC14.5.1
Use case	Content Fetching
Description	As the user selects content fetching the AXMEDIS plug-in opens and Content delivery starts. User can select the 3 different delivery modes: • **Streaming.** Similar to a broadcast experience, user acquires license and subsequently starts streaming content. Recommended only for higher bandwidth (450kb/s or above). • **Download.** After acquiring a license, the user can download the media (up to 10Mb/s encoding). Media can be viewd from the user's computer after the downloading process (can take 1-8 hours according to user access) • **Pre-Download**. User can first download content and then is prompted to purchase license. The user can check any time that the progress bar, indicating the download state, is advancing.
Actors	The Content Consumer (user) AXMEDIS plug-in
Assumptions	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.
Steps	1 The user selects delivery mode: pre-download, download, progressive download, streaming 2 The AXMEDIS plug-in opens and content delivery starts according to the delivery mode chosen by the user 3 The user opens the jobs panel where all current downloads are displayed 4 The user reads the remaining time for the end of transmission 5 The user can open the folder where the content is being received 6 The user can interrupt the reception of a given content
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	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. The use may also activate a previously purchased license while fetching content in progressive download.
Design suggestions	None.
Issues	None.

14.5.2 User Authentication Form

UCId	UC14.5.2
Use case	User Authentication Form

Description	The user will be requested to authenticate in order to start any content fetch or transaction
Actors	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)
Assumptions	The user has access to the Catalogue
Steps	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) 2 The user credentials are sent to the MediaClub for verification 3 The user waits for the server response 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
Post-conditions	The system shall have entered the next procedural step
Variations	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.
Asynchronous actions	None
Design suggestions	None.
Issues	None

14.5.3 Catalogue Content Transaction

UCId	UC14.5.3
Use case	Catalogue Content Transaction
Description	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.
Actors	The Content Consumer (user) The MediaClub Payment Gateway
Assumptions	The user has selected the Catalogue content

Steps	1 The MediaClub Payment Gateway shows to the user all billing information available including: o Price o Conditions for each selected item o Related use licence o Scope and limitations o Possible constraints 2 The MediaClub Payment Gateway asks the user to verify and accept presented terms 3 If the user accepts procedure continues otherwise is aborted and user is sent back to browsing 4 The user shall finalise billing information 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) 6 The MediaClub Payment Gateway requires clearance to the AXMEDIS Distributor for the provided payment ID. 7 If payment ID is cleared the user will be charged the cost 8 The MediaClub Payment Gateway provides the system the proper clearance and the license delivery is authorized. 9 The user receives confirmation of transaction OK on a web page 10 The user receives an email notification that transaction has been succesful 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
Post-conditions	The system shall have entered the next procedural step
Variations	
Asynchronous actions	None.
Design suggestions	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.
Issues	None.

14.5.4 Content Access

UCId	UC14.5.4
Use case	Content Access
Description	The user accesses his local cache containing several AXMEDIS Objects.
Actors	The Content Consumer (user)
Assumptions	The AXMEDIS Content is successfully received.
Steps	1 The user accesses the AXMEDIS Object for playing it 2 The AXMEDIS Object is delivered to either the AXMEDIS Viewer or the standard application (with an additional AXMEDIS plug-in) 3 The application detects if the Object needs to acquire a license 4 The application finds a pre-acquired license for the Object and play it 5 The application needs a new license for the Object and tries to contact the MediaClub.
Post-conditions	The system shall have entered the next procedural step
Variations	
Asynchronous actions	None.
Design suggestions	

Issues	None.

14.5.5 Content Preview

UCId	UC14.5.5
Use case	Content Preview
Description	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.
Actors	The Content Consumer (user)
Assumptions	The AXMEDIS Object has been integrally received.
Steps	1 The user opens the AXMEDIS Object locally stored in his local cache 2 The user browses the AXMEDIS Object, using the AXMEDIS Info associated to the Object 3 The user reaches a preview available for the Object 4 The user plays the AXMEDIS Object Preview
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None.
Design suggestions	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.
Issues	None.

14.5.6 License Acquisition

UCId	UC14.5.6
Use case	License Acquisition
Description	The user receives a license for playing the content
Actors	The Content Consumer (user)
Assumptions	The user is logged-in to the MediaClub The user has selected to play an AXMEDIS content
Steps	1 The user opens the protected part of the AXMEDIS Object 2 The Object is delivered to the application/viewer charged to open/play it 3 The Application/Viewer has an internal plug-in able to detect if the Object to open needs a license 4 The AXMEDIS Viewer, using the internal plug-in, contacts the MediaClub in a protected mode (a secure connection is established with the MediaClub) 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 6 The user receives the AXMEDIS license useful to open the protected part of the AXMEDIS Object 7 The user receives a confirmation page that license has been successfully issued 8 The user consumes the AXMEDIS Object following the rules contained in the AXMEDIS license
Post-conditions	The user plays the content
Variations	None
Asynchronous actions	None
Design suggestions	None.
Issues	Security, privacy and transparency are key requirements.

14.5.7 Multi-device license activation and back-up

UCId	UC14.5.7
Use case	Multi-device license activation and back-up
Description	The user copies some interesting content in a device other than initial PC
Actors	The Content Consumer (user)
Assumptions	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.
Steps	1 The user opens the copy/backup interface of the AXMEDIS Client plug-in 2 The user selects all Objects involved in the copy operation 3 The user specifies the device where the AXMEDIS Content has to be copied. 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
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	None
Issues	None.

14.5.8 Pre-ordering and registration for a group of students

UCId	UC14.5.8a
Use case	Pre-ordering for a group of students
Description	This case describe the specific procedure used by a “master user” (teacher) to buy content licenses for multiple other user.
Actors	The teacher
Assumptions	The teacher has to be registered in the system. The students (users) may not be registered in the system ;
Steps	1.The teacher orders the product and request N licenses 2.Through the commercial service, or a web server, or a pre-registered account etc, the teacher pays the bill 3. Pre-ordering is saved in the Pre order database, waiting for activation 4. The teacher receives an e-mail confirming its order, and containing the Activation Number
Post-conditions	
Variations	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
Asynchronous actions	None
Design suggestions	None
Issues	None.

UCId	UC14.5.8b
Use case	Automatic registering for a group of students
Description	This case describe the automatic registering of students (user) when they use the content pre-ordered by their teacher for the first time
Actors	The teacher The students

Assumptions	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
Steps	1 The teacher gives to the students the URL to access the content ; 2 The teacher gives the Activation Number for this content to the students ; 3 Through the dedicated client, the student access the URL given by the teacher 4 The system ask the student for the Activation Number for this product 5 The student enter the Activation Number 6 The dedicated client associate automatically computed user/device identification data and send them along with the Activation Number 7 The system creates an AXMEDIS user/device with the identification data 8 The system creates an AXMEDIS License corresponding to the parameters given at pre-ordering time, for the previously created user/device 9 The license is made available trough AXMEDIS for the user to be able to view the requested content
Post-conditions	
Variations	Depending on the initial parameters, the license is granted for a user, for a device or for a combination of both.
Asynchronous actions	None
Design suggestions	None
Issues	None.

14.6 Business Models

14.6.1 Rental

UCId	UC14.6.1
Use case	Rental
Description	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.
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	1 the customer chooses the content 2 he provides minimal personal information, chooses the payment 3 the distributor confirms the successful payment transaction 4 the customer downloads the content 5 after the content expiration date the content is not accessible anymore
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	(TBD)
Issues	None.

14.6.2 pay per download

UCId	UC14.6.2
Use case	pay per download

Description	The end user pays to download a content that will be seen only once.
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	1 the customer chooses the content 2 he provides minimal personal information, chooses the payment 3 the distributor confirms the successful payment transaction 4 the customer downloads the content 5 if the customer wants to access again to the content he has to replicate this procedure
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	(TBD)
Issues	None.

14.6.3 Sell-through

UCId	UC14.6.3
Use case	Sell-through
Description	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.)
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	like pay per download
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	(TBD)
Issues	None.

14.6.4 subscription

UCId	UC14.6.4
Use case	subscription
Description	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.
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	1 the customer chooses the subscription 2 provide personal information, chooses the payment system and the invoice method (paper mail) 3 the distributor confirms the subscription success and the customer can start using the service 4 the renewal is automatically done until the customer terminates the subscription
Post-conditions	The system shall have entered the next procedural step
Variations	None.

Asynchronous actions	None
Design suggestions	(TBD)
Issues	None.

14.6.5 pay per minute

UCId	UC14.6.5
Use case	pay per minute
Description	The customer is charged for the time the content is streamed
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	1 the customer chooses the content 2 he provides personal information, chooses the payment and invoice method (mail / paper) 3 the distributor confirms the successful payment transaction 4 the customer can start see/use the content 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
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	AXMEDIS has to provide information of the minutes required
Issues	None.

14.6.6 pay per Kb downloaded

UCId	UC14.6.6
Use case	Pay per Kb downloaded
Description	The customer is charged for the Kb downloaded
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	same procedure as pay per minute where the measure unit is the Kb instead of the minutes
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	AXMEDIS has to provide information of the minutes required
Issues	None.

14.6.7 pay per day

UCId	UC14.6.7
Use case	pay per day
Description	The customer is charged for the number of days he access to the content

Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	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
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	AXMEDIS has to provide information of the minutes required
Issues	None.

14.6.8 pay per credits

UCId	UC14.6.8
Use case	pay per credits
Description	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 **100 credits = 1 euro.** 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.
Actors	The Content Consumer (user)
Assumptions	See catalogue Content Transaction
Steps	1 User can purchase in advance (prepaid) credits. 2 Every time he rents or buys a media, a certain number of credits will be deducted from his account. 3 Prepaying a higher number of credits results in a volume discount for the user
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None
Design suggestions	
Issues	None.

14.6.9 Grouped licenses

UCId	UC14.6.9
Use case	Grouped Licenses
Description	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
Actors	VLE Provider, Content Provider, Users
Assumptions	VLE Provider includes AXMEDIS client into his VLE offer
Steps	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
Post-conditions	

Variations	None.
Asynchronous actions	None
Design suggestions	
Issues	Precise the concept of "domain"

14.6.10 Packaged offers

UCId	UC14.6.10
Use case	Packaged offers
Description	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
Actors	Distributor, Contents providers
Assumptions	
Steps	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
Post-conditions	
Variations	None
Asynchronous actions	None
Design suggestions	
Issues	None.

14.7 Advanced Payment methods

14.7.1 Gift Certificates

UCId	UC14.7.1
Use case	Gift Certificates
Description	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.
Actors	The Content Consumer (end user);
Assumptions	

Steps	**Definition** *CustomerA*: buys a credit for a friend *CustomerB*: is the friend who receive the gift **Procedure to Purchase a Gift Certificate** **Step 1: start** In the web site shop *customerA* clicks on a link 'Buy a gift certificate'. This link is part of the portal. **Step 2: payment details** The distributor application asks to the customer: • the amount to buy • mail address of the friend • payment details (these information are stored by the Distributor adding a PIN code) **Step 3: mail delivery to *customerB*** An application send an email to customerB providing information about the gift and how to redeem it. This text can be partially typed by customerA Instruction contains a link to the website where customerB can redeem the gift and (embedded in the link) a PIN code that will be burn once the customer redeem the gift. **Procedure to redeem the gift certificates** CustomerB 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) When customerB approves the Kart content, there is a control about the amount to pay and the value of the kart with 3 different situation: 1) gift value=value to purchase the customer sees a confirmation page + receives an email 2) gift value>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 ex. Gift certificate value = 50 € (that is the value associated to the PIN generated for that gift) customerB buys 30 € in Tiscali music club the new value of the PIN code is 20 € available for new purchases the customer sees a confirmation page reminding the credit available + receives an email with the link where to redeem the credit available 3) gift value<value of the kart the customer is required to chose a payment method to pay the difference or to come back to the kart to remove some items.

Post-conditions	The gift certificates has an expiration date The distributor application supports the possibility for the customer to check the gift certificates already available
Variations	None.
Asynchronous actions	?
Design suggestions	The gift certificates application is stand alone
Issues	None.

14.7.2 Wallet

UCId	UC14.7.2
Use case	Wallet
Description	Wallet is a payment account the customer opens with the Distributor for paying transactional services. The wallet creation needs a first deposit by the customer and can be used immediately. The wallet can be recharged with following deposits. The wallet saves the payment method used by the customer that is proposed for following deposit.
Actors	The Content Consumer (user)
Assumptions	

Steps	**Wallet creation** 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. Afterwards made the first deposit using the payment methods allowed by the distributor **Wallet ecare** The customer can: • Check the balance • Recharge • Check the statement (List of deposits, List of the purchases done) • Change the secure key • Change payment method used To access to the wallet ecare, to make payments, and to recharge the wallet, the security key is always requested. **Payments** if the customer decides to pay with wallet and the balance is NOT enough to cover the new purchase he is asked for recharge. **Wallet termination** • Expiration 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. • Termination The customer can ask to close his wallet removing the payment information. No refund applicable
Post-conditions	
Variations	None.
Asynchronous actions	
Design suggestions	
Issues	None.

15 AXMEDIS for Distribution towards Mobiles

15.1 General Assumptions and Notes to Architecture

1) The AXMEDIS enabled Comverse 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 Comverse 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 Comverse servers and components.
2) The AXMEDIS enabled Comverse 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 Comverse servers and components.
   c) A Transcoding platform including Codecs, configuration and Interface.
3) Categories:
   a) Categories are AXMEDIS objects stored in the AXMEDIS DB.
   b) Each category can contain or reference several content items (text for menus, audio for IVR, etc.).
   c) The Categories are arranged in a tree structure. Each Category has only one parent Category (with the exception of the root Category).
   d) Each Category can contain/reference an AXMEDIS Selection that defines the content referenced by the Category.

15.2 Use Cases

15.2.1 Transcoding New Content

UCId	UC15.2.1
Use case	Transcoding new Content
Description	The Transcoding Server makes newly published AXMEDIS objects available in formats required by the Comverse distribution system. The server fetches the objects, extracts and converts the content to the required formats, and publishes them as new AXMEDIS objects.
Actors	1. Transcoding Server: manages the conversion process, configuration and reporting. 2. Transcoding Plug-in: A platform (interface + codecs + configuration) for converting content format. 3. AXMEDIS Plugin: Interacts with the AXMEDIS network and platform on behalf of the Transcoding Server. 4. AXMEDIS Query Support. 5. AXEPTool: AXMEDIS server that notifies of new objects matching a Selection.
Assumptions	1. The Transcoding Server processes objects that match a well defined Selection. The criteria includes, but is not limited to, license attributes, content type (audio, video), format type (WAV, WMA, JPG), date range, etc. 2. When the Transcoding Server re/starts, the plugin requests the AXEPTool to be notified when new objects match the desired Selection.

Steps	1 The AXEPTool identifies a new match to the Active Selection. 2 The AXEPTool notifies the plugin that a new match is ready for transcoding. 3 The plugin forwards the event to the Transcoding Server. 4 The Transcoding Server initiates a query for new AXMEDIS items via the AXMEDIS plugin. 5 The plugin hits the Query Support service with a predefined selection, where the criteria are: license that agrees with the Comverse distribution system; content type and format (e.g. audio: WAV, WMA); time range; the max size of the result in rows; etc. 6 The Query Support returns a list of AXMEDIS objects to the plugin. 7 The plugin fetches all the listed objects from the AXMEDIS DB, and stores them locally, unprotected in a ready to use format (for conversion). 8 The plugin returns the list of objects and their physical location in the local storage to the Transcoding Server. 9 The Transcoding server sends the list to the Transcoding plugin and initiates the conversion sequence. 10 The Transcoding plugin converts each and every file in the list to the desired format(s) as required by the configuration. 11 When done, the Transcoding plugin returns a list of all the conversions made to the Transcoding server. 12 The Transcoding server generates a list of new AXMEDIS objects to create, with reference to the newly created files, and passes it to the AXMEDIS plugin for publication. 13 The plugin creates new AXMEDIS objects out of the converted files and publishes them to AXMEDIS DB. 14 The Transcoding server purges the local storage of the remains of original and converted files. 15 The Transcoding server writes an activity log reporting all actions made on the fetched AXMEDIS objects (fetch, convert, etc).
Post-conditions	None
Variations	1. The transcoding procedure is typically triggered by a new-match notification from the AXEPTool, but can also be triggered by a scheduled event. 2. Selection of objects to convert may be added with the list of objects that previously failed transcoding X times or expired by T time-range. 3. The OID of AXMEDIS objects that failed transcoding for X times or expired by T time-range will be added to a Rejected list.
Asynchronous actions	None
Design suggestions	None
Issues	1. Transcoding a large number of items may take a long time (e.g. hours, days). Meanwhile, the properties and attributes of the object (e.g. license) may change, making the conversion irrelevant or illegal.

15.2.2 The APS Loads the Content Tree

UCId	UC15.2.2
Use case	The APS Loads Content Tree.
Description	The Comverse APS loads the tree of Categories which subscribers browse in search of content, and the content that is associated with the Categories.

Actors	1. APS (Comverse Application Server) 2. AXMEDIS plugin: Interacts with the AXMEDIS network and platform on behalf of the APS. 3. AXMEDIS node: An AXMEDIS network node storing all required objects and providing AXMEDIS services with online (high, immediate) availability. 4. AXMEDIS Query Support.
Assumptions	1. The root Category OID is known to the APS.
Steps	1 The APS calls the AXMEDIS plugin to load the content tree, beginning with the root Category OID. 2 The plugin retrieves all the Category objects to form the Categories tree from the local AXMEDIS node DB. 3 The plugin calls the Query Support with the Selection of each Category. 4 The Query support returns the result for each selection. 5 The plugin returns a complete content tree – categories and content listing – to the APS. 6 The Plugin removes objects that are not stored on the local AXMEDIS node from the content list (as they are not available).
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	1. (relevant only if step 6 is not applicable) The result of the selections may reference objects that are not yet stored on the local AXMEDIS node – they will not be available for immediate use by the APS.

15.2.3 Subscriber Browses the Content Tree

UCId	UC15.2.3
Use case	The Subscriber browses the content tree.
Description	The Subscriber begins browsing the Category tree for content. The system modifies the content tree according to the subscriber's previous preferences, handset capabilities and DRM rules.
Actors	1. Subscriber – End consumer. 2. APS (Comverse Application Server) 3. PE (Comverse Personalization Engine) 4. HME (Comverse Handset Management Engine)
Assumptions	None

Steps	1 The Subscriber starts browsing for content via a UI (Web, WAP, IVR, etc). 2 The APS creates a copy of the content tree to be modified according to subscriber's profile. 3 If the destination device type is known, then 3.1 The APS sends the content tree to the HME. 3.2 The HME removes from the content tree items that: 3.2.1 The Subscriber's handset does not support 3.2.2 Their license prohibits distribution to the Subscriber's handset or handset type. 3.3 The HME returns the filtered content tree to the APS. 4 The APS sends the content tree to the PE. 5 The PE loads the *Subscriber's* profile, and rearranges the content tree according to his/her preferences. (Suggested: favorites and last purchased artists/genre, UI language, sort tree by, sort order, removal of items for limited UIs (e.g. IVR), etc. 6 The PE returns the personalized content tree to the APS. 7 The Subscriber browses the personalized content tree. 8 The APS monitors the Subscribers browsing behavior and data to be added to the user browsing.
Post-conditions	None
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

15.2.4 The Subscriber Samples Content

UCId	UC15.2.4
Use case	The Subscriber samples content.
Description	The Subscriber requests to sample content on his/her handset.
Actors	1. Subscriber 2. APS
Assumptions	1. The Subscriber can only sample audio content.
Steps	1 The Subscriber requests to sample a specific item from the content list. 2 The APS verifies that the relevant sample file is available and ready. 3 The APS loads and pushes the sample file to the subscriber's handset.
Post-conditions	
Variations	1. If the handset type is not determined, then the sample with optimal/lowest quality (configurable) is played. 2. If the sample is not available, the APS alerts the Subscriber that a sample is not available.
Asynchronous actions	None
Design suggestions	None

Issues	1. It is possible that some media will be playable in the destination handset, but not available for the subscriber's handset. The Subscriber may browse content that he/she would not be able to sample before purchasing for another person. 2. The subscriber's sample may be at higher quality than that which the destination handset can handle. It may mislead the subscriber to buy a low quality item although the sample was at high quality. In this case the APS must notify the Subscriber of the issue. 3. If the object's license dis/allows actions on specific handset types, it must also specify what is dis/allowed when the handset type is not determined - a "general handset" scenario. If there are no specifications for general handset, what actions are dis/allowed by default?

15.2.5 The Subscriber Purchases Content

UCId	UC15.2.5
Use case	The Subscriber purchases content.
Description	The Subscriber purchases content.
Actors	1. Subscriber. 2. APS. 3. HME.
Assumptions	
Steps	1 The Subscriber requests to sample a specific item from the content list. 2 The Subscriber provides the MSISDN (ID) of the destination device. 3 The APS calls the HME to determine the type of the destination device. 4 The APS verifies that the purchase transaction is applicable, considering: 4.1 Availability of the content file in the format supported by the Handset. 4.2 Subscriber's status and credit. 4.3 DRM allowing distribution to the handset. 5 The APS pushes the content to the destination device.
Post-conditions	None
Variations	1. If the HME is unable to determine the destination device type, then the Subscriber can provide it.
Asynchronous actions	None
Design suggestions	None
Issues	1. If the object's license dis/allows actions on specific handset types, it must also specify what is dis/allowed when the handset type is not determined - a "general handset" scenario. If there are no specifications for general handset, what actions are dis/allowed by default?

16 AXMEDIS for Distribution towards i-TV

16.1 User Terminal Installation and Configuration

16.1.1 User Hardware Installation

UCId	UC16.1.1
Use case	User Hardware Installation
Description	The user installs the required hardware in his PC
Actors	The Content Consumer (user)

Assumptions	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.
Steps	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 2 The user physically installs the DVB-IP adapter according to the installation instructions provided by the manufacturer 3 The user connects the DVB-IP adapter to the satellite dish 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 5 The user configures the DVB-IP adapter according to instructions 6 The user checks that the satellite signal is received correctly
Post-conditions	The system shall have entered the next procedural step
Variations	Some DVB-IP cards might not support all AXMEDIS functionalities. It is recommended that the user obtains a supported DVB-IP adapter.
Asynchronous actions	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.
Design suggestions	A list of compatible adapters should be prepared. Full installation instructions should be given to the user.
Issues	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.

16.1.2 User Software Installation

UCId	UC16.1.12
Use case	User Software Installation
Description	The user installs the AXMEDIS Client Application
Actors	The Content Consumer (user)
Assumptions	The DVB-IP hardware is correctly installed and configured. The user PC has a working connection to the Internet.
Steps	1 The user obtains the AXMEDIS Client Application Setup (e.g., from the Internet, from a CD, ...) 2 The user runs the AXMEDIS Client Application Setup 3 The user follows the steps of the installation 4 Some information entered by the user is used to create his profile, that is securely communicated via Internet to a server
Post-conditions	The system shall have entered the next procedural step
Variations	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).
Asynchronous actions	None.

Design suggestions	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.
Issues	None.

16.1.3 User Registration

UCId	UC16.1.3
Use case	User Registration
Description	The user registers himself in order to access the AXMEDIS service
Actors	The Content Consumer (user)
Assumptions	The user has successfully installed the hardware and software AXMEDIS components.
Steps	1 The user runs the AXMEDIS Client Application registration procedure 2 The user enters or updates his personal profile, that is securely stored on the Server 3 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.) 4 The AXMEDIS Client Application may update its internal state by receiving appropriate files from the Server (e.g., group memberships)
Post-conditions	The user is ready to use the AXMEDIS service and access the published Content. (Access can be restricted only to some components)
Variations	The user may re-run the procedure to update his profile. In some situations this procedure could be automatic and hidden to the user. 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.
Asynchronous actions	None.
Design suggestions	The Server shall store the user profile and use it to personalize content presentation, messages, etc.
Issues	None.

16.1.3.1 Application Selection

UCId	UC16.1.3.1
Use case	Application Selection
Description	The user selects the preferred Application model
Actors	The Content Consumer (user)
Assumptions	The user has successfully registered himself in order to fully access the AXMEDIS service.
Steps	1 The user runs the AXMEDIS Client Application 2 The user select the desired Application model between the three available: o Standard application o Cache-based Distribution on i-TV o Cached-based Personalised Content Distribution
Post-conditions	None
Variations	The user can change the selected Application with no restrictions
Asynchronous actions	None

Design suggestions	The Application selection should be easy and fast to perform
Issues	None

16.1.3.2 User Profiling

UCId	UC16.1.3.2
Use case	User Profiling
Description	The user provides his/her preferences about AXMEDIS contents
Actors	The Content Consumer (user)
Assumptions	The user has successfully registered himself in order to fully access the AXMEDIS service.
Steps	1 The user runs the AXMEDIS Client Application User Profiling procedure 2 The user provides or updates his/her preferences about AXMEDIS contents 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 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 5 The user saves the his/her User Profile 6 The user connects to the Internet to update his/her User Profile stored in the Server
Post-conditions	None
Variations	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.
Asynchronous actions	None
Design suggestions	None
Issues	None

16.2 Content Listing

16.2.1 Content Web Listing

UCId	UC16.2.1
Use case	Content Web Listing
Description	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).
Actors	The Content Consumer (user)
Assumptions	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.

Steps	1 The user reaches the AXMEDIS Content Web List 2 The user displays the proposed content using different criteria (type, author, content producer, production date) 3 The user inserts some key words for filtering Object potentially interesting for him 4 The user reads all available information (contained in the AXMEDIS Info) associated to the AXMEDIS Object, helpful for voting 5 The user plays, by downloading needed data, some short previews (if this option is available)
Post-conditions	The system shall have entered the next procedural step (Content Voting)
Variations	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).
Asynchronous actions	None.
Design suggestions	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.
Issues	None.

16.2.2 Content Carousel Listing

UCId	UC16.2.2
Use case	Content Carousel Listing
Description	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.
Actors	The Content Consumer (user).
Assumptions	The user knows the type of search that he wants to start (generic, advanced, personalized, pre-stored, etc.)
Steps	1 The user uses some pre-defined functionalities to filter the content 2 The user applies its own profile (locally stored) to the AXMEDIS offer to best match his interest in the offered content 3 The user enters some key words in the content browsing 4 The user reads all available information (contained in the AXMEDIS Info) associated to the AXMEDIS Object, helpful for selection 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.
Post-conditions	The system shall have entered the next procedural step (Content Selection)
Variations	None
Asynchronous actions	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.
Design suggestions	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.
Issues	None.

16.3 Content Voting

UCId	UC16.3
Use case	Content Voting
Description	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.
Actors	The Content Consumer (user)
Assumptions	The user still has available preferences to vote Content in the proposed web list.
Steps	1 The user chooses one or more AXMEDIS Objects he wishes to receive by push inside the AXMEDIS Carousel 2 The user sends his preferences to the server side 3 The user receives a receipt about his vote expression 4 The user receives a notification saying if his AXMEDIS voted Object has entered in the AXMEDIS Carousel
Post-conditions	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.
Variations	None.
Asynchronous actions	The voting action could affect the user profile. A synchronisation between the local stored and the server stored user profiles could be done.
Design suggestions	None.
Issues	None.

16.4 Content Selection

16.4.1 Manual Content Selection

UCId	UC16.4.1
Use case	Manual Content Selection
Description	The user selects (manually) the scheduled content that will be received at the indicated time by push.
Actors	The Content Consumer (user)
Assumptions	The user leaves the computer and the AXMEDIS Client Application turned on during the time window of the selected transmission.
Steps	1 The user double clicks on the AXMEDIS Object in order to select it for reception 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
Post-conditions	The system shall have entered the next procedural step
Variations	The user could select his AXMEDIS Object from a remote computer and the order could be passed to his local AXMEDIS Client Application.
Asynchronous actions	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.
Design suggestions	None.
Issues	None.

16.4.2 Automatic Content Selection

UCId	UC16.4.2
Use case	Automatic Content Selection

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

16.5 Content Reception

UCId	UC16.5
Use case	Content Reception
Description	The user can check any time that the progress-bar, indicating the download state, is advancing.
Actors	The Content Consumer (user)
Assumptions	The user has selected, either manually or automatically, an AXMEDIS Object distributed in the AXMEDIS Carousel.
Steps	1 The user opens the jobs panel where all current downloads are displayed 2 The user reads the remaining time for the end of transmission 3 The user can open the folder where the content is being received 4 The user can interrupt the reception of a given content
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	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.
Design suggestions	None.
Issues	None.

16.6 Content Reparation

UCId	UC16.6
Use case	Content Reparation
Description	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.
Actors	The Content Consumer (user)
Assumptions	The user has an Internet Connection, needed to contact the server in order to repair the incomplete AXMEDIS Object.

Steps	1 The user tries to open an AXMEDIS Object from the access panel provided from the Client Application Interface 2 The user receives a pop-up saying that some packets were lost during the multicast transmission 3 The user decides either to repair the Object via unicast or to wait for a next retransmission or to delete the incomplete Object.
Post-conditions	The system shall have entered the next procedural step
Variations	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. 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.
Asynchronous actions	The user tries to repair an AXMEDIS Object too old.The AXMEDIS Object could be not more available for repairing.
Design suggestions	Definition of a strategy to combine Push/Pull technologies.
Issues	None.

16.7 Content Access

UCId	UC16.7
Use case	Content Access
Description	The user accesses his local cache containing several AXMEDIS Objects.
Actors	The Content Consumer (user)
Assumptions	The AXMEDIS Content is successfully received.
Steps	1 The user accesses the AXMEDIS Object for playing it 2 The AXMEDIS Object is delivered to either the AXMEDIS Viewer or the standard application (with an additional AXMEDIS plug-in) 3 The application detects if the Object needs to acquire a license 4 The application finds a pre-acquired license for the Object and play it 5 The application needs a new license for the Object and tries to contact the AXCS. 6 The user asks corresponding PMS for authorisation of operation 7 The PMS sends authorisation result to the user, and keys for unprotecting them, if needed
Post-conditions	The system shall have entered the next procedural step
Variations	If the user is not authorised because she has no license, then it contacts PMS for acquiring it.
Asynchronous actions	None.
Design suggestions	
Issues	None.

16.8 Content Preview

UCId	UC16.8
Use case	Content Preview

Description	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.
Actors	The Content Consumer (user)
Assumptions	The AXMEDIS Object has been integrally received.
Steps	1 The user opens the AXMEDIS Object locally stored in his local cache 2 The user browses the AXMEDIS Object, using the AXMEDIS Info associated to the Object 3 The user reaches a preview available for the Object 4 The user plays the AXMEDIS Object Preview
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None.
Design suggestions	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.
Issues	None.

16.9 License Acquisition

UCId	UC16.9
Use case	License Acquisition
Description	A user tries to acquire a license for consuming a protected and governed AXMEDIS Object.
Actors	The Content Consumer (user)
Assumptions	The user should have an Internet connection (well functioning) in order to reach the AXCS and obtain the license for playing an AXMEDIS Object. 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. The user station should support all secure protocols.
Steps	1 The user wants to acquire an AXMEDIS object to play the protected resources within it 2 The AXMEDIS Object is delivered to the application/viewer charged to open/play it 3 As the AXMEDIS Object is protected and governed, the user is informed that he has to purchase the appropriate license for playing it 4 The user requests the appropriate license 5 The license server generates the license with the relevant parameters (principal, right/s, resource and conditions) and validates the license 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. 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.
Post-conditions	The user shall respect the rules contained in the received authorization.
Variations	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. 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)

Asynchronous actions	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.
Design suggestions	None.
Issues	E-commerce backend and transactional functionalities should be available and in place. Security, privacy and transparency should be some fundamental basis.

16.9.1 User Identification

UCId	UC16.9.1
Use case	User Identification
Description	The user will be requested to identify and provide credentials needed to ensure that the requested transaction (purchase/rental) is valid and legal.
Actors	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)
Assumptions	See License Acquisition Use Case.
Steps	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) 2 The user credentials are sent to the AXCS for verification 3 The user waits for the server response 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
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	See License Acquisition.
Design suggestions	None.
Issues	See License Acquisition.

16.9.2 Billing

UCId	UC16.9.2
Use case	Billing
Description	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.
Actors	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)
Assumptions	See License Acquisition Use Case.

Steps	1 The AXCS shows to the user all billing information available including: o Price o Conditions for each selected item o Related use licence o Scope and limitations o Possible constraints 2 The AXCS asks the user to verify and accept presented terms 3 If the user accepts procedure continues otherwise is aborted and user is sent back to browsing 4 The user shall finalise billing information 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) 6 The AXCS requires clearance to the AXMEDIS Distributor for the provided payment ID. 7 If payment ID is cleared the user will be charged the cost 8 The AXCS provides the system the proper clearance and the license delivery starts.
Post-conditions	The system shall have entered the next procedural step
Variations	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.
Asynchronous actions	See License Acquisition
Design suggestions	None.
Issues	See License Acquisition.

16.10 Content Backup

UCId	UC16.10
Use case	Content Backup
Description	The user copies some interesting content in a backup support (either external or internal).
Actors	The Content Consumer (user)
Assumptions	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). The Backup Operation has to be expressly authorized in the license terms.
Steps	6 The user opens the backup interface of the AXMEDIS Client Application 7 The user selects all Objects involved in the backup operation 8 The user specifies the backup unit where the AXMEDIS Content has to be copied.
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	The AXMEDIS Content could be automatically deleted during the backup operation.
Design suggestions	The Backup operation should interact with the Intelligent Cache Manager before starting.
Issues	None.

16.11 Content Restore

UCId	UC16.11
Use case	Content Restore

Description	The user restores some previously backup AXMEDIS Object from the backup support (either external or internal).
Actors	The Content Consumer (user)
Assumptions	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.
Steps	1 The user opens the restore interface of the AXMEDIS Client Application 2 The user selects all Objects involved in the restore operation 3 The user specifies the backup unit where the AXMEDIS Content from which the Content has to be restored.
Post-conditions	The system shall have entered the next procedural step
Variations	None.
Asynchronous actions	None.
Design suggestions	The Restore operation should interact with the Intelligent Cache Manager after restoring.
Issues	None.

16.11.1 Cache Preloading

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

16.12 Cache Cleaning

UCId	UC16.12
Use case	Cache Cleaning
Description	The user emptiesthe local cache
Actors	The Content Consumer (user)
Assumptions	The Cache has some AXMEDIS objects
Steps	1 The user runs the AXMEDIS Client Set Up application 2 The user activates the Empty Cache functionality 3 The user may immediately asks for a new Cache Preloading
Post-conditions	
Variations	
Asynchronous actions	None.
Design suggestions	
Issues	None.

16.13 Cache-Based Personalised Content Distribution specific Use Cases

16.13.1 Automatic Content Access Set Up

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

16.13.2 AXMEDIS Channel personalisation

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

16.13.3 Automatic Content Access

UCId	UC16.13.3
Use case	Content Access
Description	The user plays an AXMEDIS Channel
Actors	The Content Consumer (user)
Assumptions	The user has already made the Automatic Content Access set up

Steps	1 The user switches on the AXMEDIS Client Application 2 The user selects the Cache-Based Personalised Content Distribution Application 3 The user selects an AXMEDIS channel 4 The user plays the AXMEDIS channel
Post-conditions	
Variations	
Asynchronous actions	None.
Design suggestions	
Issues	None.

16.13.4 AXMEDIS Channel PVR functionalities

UCId	UC16.13.4
Use case	Content Access
Description	The user activates the PVR functionalities graphic interface
Actors	The Content Consumer (user)
Assumptions	The user has already made the Automatic Content Access set up
Steps	1 The user switches on the AXMEDIS Client Application 2 The user selects the Cache-Based Personalised Content Distribution Application The user activates the PVR functionalities: • play • fast forward • rewind • record (the user records some content from an AXMEDIS channel to the hard disk) • pause
Post-conditions	
Variations	
Asynchronous actions	None.
Design suggestions	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).
Issues	None.

17 AXMEDIS for Distribution to PDA via Kiosks

17.1 Content Catalogue Creation

UCId	UC17.1
Use case	Content Catalogue Creation
Description	The kiosk manager creates a catalogue
Actors	The kiosk manager, AXMEDIS Content Production
Assumptions	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)

Steps	1 The kiosk manager logs into the system. 2 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: ***Select From AXEPTool (Where {Key1=XYZ, Key2=KKK…)*** 3 The query support system returns a AXOID list 4 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. 5 The kiosk manager performs a new query to retrieve the wanted objects: ***Get From AXEPTool (Where {AXOID1=X, AXOID2=Y…)*** 6 The query support system returns a AXMEDIS objects list 7 The kiosk manager checks with the Composition Rules Editor presently available composition rules 8 If available rules are adequate proceeds to next step 9 If available rules need to be modified or are lacking then modifies / defines Composition Rules 10 The kiosk manager checks with the Formatting Rules Editor presently available formatting rules 11 If available rules are adequate proceeds to next step 12 If available rules need to be modified or are lacking then modifies / defines Formatting Rules 13 The kiosk manager activates the Composition Engine that creates a new AXMEDIS object starting from the collected objects 14 The kiosk manager activates the Formatting Engine that creates a new AXMEDIS object
Post-conditions	The system shall have returned in operational mode
Variations	The procedure may be launched remotely from the kiosk management server or on a timed base exploiting all functionalities of the composition, formatting and publication engines
Asynchronous actions	None
Design suggestions	None
Issues	Backend and transfer functionalities should be available and in place

17.2 Content Catalogue Loading (publication)

UCId	UC17.2
Use case	Content Catalogue Loading (publication)
Description	The kiosk manager publishes a catalogue
Actors	The kiosk manager, local AXDB, AXMEDIS Publication environments
Assumptions	The kiosk manager is a registered AXMEDIS user with a specific UID and has all the right and tools to perform the operation

Steps	1 The kiosk manager logs into the system. 2 The kiosk manager checks with the Publication Rules Editor presently available publication rules 2.a If available rules are adequate proceeds to next step 2.b If available rules need to be modified or are lacking then modifies / defines Publication Rules 3 The kiosk manager verifies clients view profiles and eventually updates them 4 The kiosk manager activates the Publication Engine that uses the Publication Rules defined in conjunction with the Client View Profiles to access the local AXDB (end eventually the Formatting Engine) to feed the Distributor server with the catalogue and top ten AXMEDIS object to be distributed
Post-conditions	The system shall have returned in operational mode
Variations	The procedure may be launched remotely from the kiosk management server
Asynchronous actions	None
Design suggestions	None
Issues	Backend and transfer functionalities should be available and in place

17.3 Content Catalogue Loading Update

UCId	UC17.3
Use case	Content Catalogue Loading Update
Description	The kiosk manager logs into the system and loads the new version of the content list that will enable the end-user to select and buy AXMEDIS developed/delivered content thanks to browsing and previewing.
Actors	The kiosk manager accessing the kiosk starts the application that loads locally to the kiosk the presently available content catalogue and launches the content updating interface & procedures.
Assumptions	The kiosk should be connected to the backend and all components should be well functioning. Data transfer will be operated via FTP on a scheduled basis with an expected change rate of 3-4 weeks.
Steps	1 The kiosk manager logs into the system. 2 The kiosk manager perform a “switch to maintenance mode” for the target kiosk 3 The target kiosk system exits normal operational and enters in maintenance mode. 4 The application front-end loads all system maintenance application. 5 The kiosk manager launches the catalogue upload procedure which: 5.a Contacts the Distribution Server requesting the send of the catalogue 5.b Receives the catalogue 5.c The procedure extracts from the catalogue the list of the top ten AXMEDIS objects 6 The kiosk manager activate the Content Top ten check procedure 6.a The procedure retrieves the list o the top ten content 6.b The procedure automatically removes from local storage all AXMEDIS objects out of the new top ten list 6.c Contacts the Distribution Server requesting the send of the specified list of AXMEDIS objects 6.d Receives the requested AXMEDIS objects 6.e Once new top ten AXMEDIS objects are received locally the local storage is updated and the procedure ends 7 The kiosk manager launches the local system check procedure to verify if all is in order 8 The kiosk manager exits the maintenance mode

Post-conditions	The system shall have returned in operational mode
Variations	The procedure may be managed by satellite (see later on) and is foreseen to handle problems in the data load & publication
Asynchronous actions	None
Design suggestions	None
Issues	Backend and transfer functionalities should be available and in place

17.4 Kiosk start-up

UCId	UC17.4
Use case	Kiosk start-up
Description	The system is starting up and loads applications & data that will enable the end-user to select and buy AXMEDIS developed/delivered content thanks to browsing and previewing.
Actors	The end user accessing the kiosk starts the application that loads locally to the kiosk the presently available content catalogue and launches the content browsing & previewing interface.
Assumptions	The kiosk should be connected to the backend and all components should be well functioning. Anomalies in functioning should be classified and checked in order to inhibit functioning when operational conditions will not ensure proper kiosk / service functioning
Steps	1 The system at start-up shall load the application front-end. 2 The application front-end loads all application modules and performs a full system check encompassing: 2.a Verify network connectivity 2.b Verify backend availability 2.c Verify local appliances functionality 3 Depending on system check results the system performs what follows: 3.a The system is ready to be used or 3.b signalling out of service condition
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	e-commerce backend and transactional functionalities should be available and in place

17.5 User registration to kiosk

UCId	UC17.5
Use case	User registration to kiosk
Description	The system is fully functional and end-users interacts with the system to register and access the kiosk application
Actors	The end user accessing the kiosk, the kiosk authentication application
Assumptions	As in previous case

Steps	1 The system presents the user a registration form with the following data: o First Name [mandatory], o Last Name [mandatory], o Age [mandatory], o Address [mandatory]: ▪ mail address, phone, mobile, e-mail, VAT code ▪ … o Default Language [mandatory], o Preferred payment method: ▪ pre-paid-cards, credit card ▪ … this is a set of optional data to be loaded if the customer wants the info to be permanently offered as default solution), o Payment method: ▪ card #, validity from, validity to, type ▪ … this is a set of optional data to be inserted only if the user is willing to directly acquire a service from the kiosk or if the kiosk service is considered to be a pay per use service), o Billing info: ▪ mail address, phone, mobile, e-mail, ▪ VAT code ▪ … this data has to be provided if different from the one in the address data section, if left empty billing system will default values from user address), o Preferred device: o PDA, o Smart phone, o Other 2 The user provides the required data 3 The user confirms input operation ending either pressing a button on the interface or any other widget. 4 The kiosk performs a check on data provided to verify completeness and correctness (as far as possible like for e-mail formats or number of digits for a VAT code or credit card…) 5 Depending on check results the system performs either operation: 5.1 Requires the user to re-input/correct data or add missing mandatory items 5.2 The kiosk presents the user a filled in form to require data confirmation or change 6 The user modifies or confirms provided data (in case modification apply steps 1-5 have to be re-iterated) 7 The kiosk local application server properly formats the data and send a request to the AXMEDIS Registration Service 8 The kiosks prompts the user to wait for registration clearance 9 In case of success the AXMEDIS Registration Service sends back to the kiosk user final UID 10 The kiosks retrieves the registration clearance, informs the user of performed registration, stores provided UID and sends the confirmation e-mail to the user specified account
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None

Issues	In the kiosk scenario the case of a user registering for the 1st 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.

17.6 User login

UCId	UC17.6
Use case	User login
Description	The system is fully functional and end-users interacts with the system to register and access the kiosk application
Actors	The end user accessing the kiosk, the user device (PDA or mobile) and the kiosk authentication application
Assumptions	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
Steps	1 The user interacts with the application front-end (selecting registration from language selection case) 2 The application front end invokes the local user login 3 The kiosk user management sends to the application front end the user data structure to be filled 4 The application front end asks the user to provide the login data (UID) 5 The user inserts the data and confirms. 6 Filled in data structure is sent back to the kiosk user management 7 The kiosk user management checks user information locally 8 The kiosk user management retrieves user related data via UID (in case the UID is not present the user will be requested to register) 9 The kiosk user management sends user data to the AXCS for verification (via AXCS web service interface) 10 The AXCS checks received info 11 The AXCS logs the registration event 12 The AXCS sends back to the kiosk user management a ACK 13 The kiosk user management confirms the login to the application front end 14 The application front end grants access to available services: application front-end presents the user a screen with the possible activities 14.a Browse the catalogue 14.b Modify own data 14.c View support information 14.d Logout
Post-conditions	The system shall have entered the next procedural step
Variations	NA
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	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 beheave as previously specified.

17.7 Content Browsing & Previewing

UCId	UC17.7
Use case	Content Browsing & Previewing
Description	The system is fully functional and end-users can browse and preview the content listed in order to select and buy AXMEDIS developed/delivered.
Actors	The end user uses the kiosk application to brows and preview the presently available content listed in the catalogue and eventually launches the selection process
Assumptions	As in previous case
Steps	1 The system presents the content list 2 The end user browses the list 3 The end user selects an item 4 The end user asks for content preview 5 Depending on content format a preview is presented as follows: 5.a Brief description for text 5.b Thumbnail for images 5.c X sec sample for Audio (X will depend on IPR rules) 5.d X sec sample for Video (X will depend on IPR rules) 5.e X sec sample for Animations (X will depend on IPR rules) 5.f X sec sample for Multimedia (X will depend on IPR rules) 6 The end users decides next step between: 6.a Activate acquiring procedure 6.b Returning to browsing
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.8 Content Selection and Chart Management

UCId	UC17.8
Use case	Content Selection And Chart Management
Description	The end user selects a content and either proceeds to check out or goes back browsing
Actors	The end user operates selections that condition the check out process
Assumptions	As in previous case
Steps	1 The end user selects a specific content (it could be both in browsing or previewing mode) 2 The user requests content addition to the chart 3 The user requests to proceed either to check out or to continue browsing 4 Depending on previous step results the system enters one of the following to states: 4.a Check out procedure activation 4.b Browsing & previewing mode
Post-conditions	The system shall have entered the next procedural step
Variations	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.
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode

Design suggestions	None
Issues	As in previous case

17.9 Check Out Procedure Initiation

UCId	UC17.9
Use case	Check Out Procedure Initiation
Description	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
Actors	The end user that is finalising the check out procedure
Assumptions	As in previous case
Steps	1 The system enters protected mode 2 A secure connection is established with the certification authority
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.10 Purchasing / Acquiring / Renting

UCId	UC17.10
Use case	Purchasing / Acquiring / Renting
Description	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)
Actors	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.
Assumptions	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.

Steps	1 The system present 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.
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.11 Repository Selection

UCId	UC17.11
Use case	Repository Selection
Description	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
Actors	The local system, AXMEDIS Certification Authority and any remote AXMEDIS database belonging to the federation and reachable form the local system
Assumptions	As in previous case
Steps	1 The system checks each selected item for local / remote availability 2 In case of remote availability a secure channel is established and data cashed locally
Post-conditions	The system shall have entered the next procedural step (in case of remotely available content, local cashed content will have to be removed)
Variations	NA
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.12 Destination Target Identification (Unique Id For Target – Wifi)

UCId	UC17.12
Use case	Destination Target Identification (Unique Id For Target – Wifi)

Description	The system has to identify the end delivery platform and, via a unique ID generate the required licence for fruition.
Actors	The local system and the PDA / mobile device
Assumptions	As in previous case.
Steps	1 The system identifies the end-user device and extracts a unique ID 2
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	none
Issues	As in previous case

17.13 Delivery Template Selection (Depending On Device)

UCId	UC17.13
Use case	Delivery Template Selection (Depending On Device)
Description	The system elects the proper delivery template for the end use device
Actors	The local system, the end user fruition device
Assumptions	As in previous case
Steps	1 The system identifies the class of delivery device 2 The system selects the template to be used for delivery (for example if the viewer has to be downloaded with the object…)
Post-conditions	The system shall have entered the next procedural step
Variations	In case a cross channel delivery is requested it will be necessary to access to additional info in order to properly secure delivery and fruition process on a device that is not available at transaction time
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.14 Delivery Format Selection (Depending On Content)

UCId	UC17.14
Use case	Delivery Format Selection (Depending On Content)
Description	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.
Actors	The local system, the end user fruition device
Assumptions	As in previous case
Steps	1 Based on the end-user device identification and delivery template the system selects the delivery format 2 The system verifies if the availability of all components 3 The system combines the required components of the delivery template to complete the delivery format 4 The system starts the preliminary checks necessary to ensure proper delivery
Post-conditions	The system shall have entered the next procedural step
Variations	In case a cross channel delivery is requested it will be necessary to ensure that safe delivery conditions can be met

Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.15 Device Compatibility (Roll Back In Case Of Failure)

UCId	UC17.15
Use case	Device Compatibility (Roll Back In Case Of Failure)
Description	The system checks that the end-use device and the delivery are not blocked by some factor that does not depend by delivery template or format and performs (eventually) necessary recovery actions
Actors	The local system, the end user fruition device, the AXMEDIS certification authority
Assumptions	As in previous case
Steps	1 Given the combination of selected content, fruition device, delivery template and format the systems performs a final consistency check 2 According to check results the system proceeds either: 2.a In the delivery process or 2.b Performs a roll back request (including billing cancelling and money refund)
Post-conditions	The system shall have entered the next procedural step
Variations	In case a cross channel delivery is requested it will be necessary to ensure that safe delivery conditions can be met. A customer feedback will be necessary within a 8 day from transaction execution date. Timestamps for this should be provided to the customer and stored both locally and at the AXMEDIS certification authority
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.16 Storage Availability (Roll Back In Case Of Failure)

UCId	UC17.16
Use case	Storage Availability (Roll Back In Case Of Failure)
Description	The system should verify that the final destination device has enough storage room to accommodate the selected object delivery.
Actors	The local system, the end user fruition device
Assumptions	As in previous case
Steps	1 The system checks if the end-fruition device has enough storage to host the selected content 2 According to check results the system proceeds either: 2.a In the delivery process or 2.b Performs a roll back request (including billing cancelling and money refund)
Post-conditions	The system shall have entered the next procedural step
Variations	In case a cross channel delivery is requested it will be necessary to ensure that safe delivery conditions can be met
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.17 Billing

UCId	UC17.17
Use case	Billing
Description	The system finalises the purchase / acquisition process and produces the billing related information (this step is performed in parallel to the delivery)
Actors	The local system, the AXMEDIS certification authority and the end user fruition device
Assumptions	As in previous case
Steps	1 The system formalises the economic transaction into a proper bill 2 The system sends the billing info to the end-user (according to provided billing info) 3 The system sends the billing info to the AXMEDIS certification authority for the required subsequent processing steps
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.18 Data Delivery

UCId	UC17.18
Use case	Data Delivery
Description	The selected data is loaded onto the end-user device (this step is performed in parallel to the billing one)
Actors	The local system, the end user fruition device, the customer
Assumptions	As in previous case
Steps	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
Post-conditions	The system shall have entered the next procedural step
Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.19 Check Out Procedure Closure

UCId	UC17.19
Use case	Check Out Procedure Closure
Description	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)
Actors	The system or the end user that is finalising the check out procedure
Assumptions	As in previous case
Steps	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
Post-conditions	The system shall have entered the next procedural step

Variations	None
Asynchronous actions	Periodic system checks should be performed and in case of negative result system should be placed in NON OPERATIONAL mode
Design suggestions	None
Issues	As in previous case

17.20 Successful Delivery Check (Recovery In Case Of Failure)

UCId	UC17.20
Use case	Successful Delivery Check (Recovery In Case Of Failure)
Description	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
Actors	The local system, the end user fruition device, the customer
Assumptions	As in previous case
Steps	1 The local system should monitor the download process to ensure a smooth delivery 2 The kiosk delivery module identifies the target device 3 Depending on target device the Kiosk delivery module acts as follows: 3.a The target device is a terminal (POP) 3.a.1 The Kiosk delivery module adapts the content to the fruition device (if necessary) 3.a.2 The Kiosk delivery module returns to the application front end the info needed to retrieve the locally cached AXMEDIS object(s) 3.a.3 The application front end loads a page to confirm delivery and grant access to the AXMEDIS object(s) 3.b The target device is a user PDA 3.b.1 Adapts the content to the fruition device (if necessary) 3.b.2 The Kiosk delivery module retrieves device data (kind, storage, certificate...) 3.b.3 The Kiosk delivery module performs required check on received device data 3.b.4 If checks are positive the Kiosk delivery module loads a page to ask downlad activation 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) 3.b.6 The kiosk delivery module takes the cached content from the local storage 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:

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

17.21 Content fruition after download on PDA or Mobile

UCId	UC17.21
Use case	Content fruition after download
Description	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
Actors	The local system, the AXOM, the domain PMS, the AXMEDIS Certification Supervisor, the end user and the fruition device
Assumptions	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
Steps	1 The user requests access to the downloaded content 2 The local viewer gets the license from the governed object 3 The local viewer gets the AXOID from the governed object 4 The local viewer gets the UID 5 The local viewer gets the device ID 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 7 The viewer informs the user of being performing a licensing check and enters a wait state for either the keys or a NACK 8 The domain PMS requires to the AXMEDIS Certification Supervisor to perform the check and if positive generate the related user keys 9 The domain PMS waits for the either the keys or a NACK 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: 10.a Sends back to the requesting PMS domain needed user keys (in case of positive result) 10.b Sends back to the requesting PMS domain a NACK 11 The PMS domain receives the reply and forwards it to the requesting viewer (via AXOM) 12 Depending on check results the viewer proceeds as follows: 12.a Allows content fruition 12.b Blocks content fruition
Post-conditions	The user is free to perform further requests on the object, return to browse the catalogue, acquire new objects and / or leave the system.
Variations	none
Asynchronous actions	None
Design suggestions	None
Issues	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

17.22 Client Based Content License Verification (Access Deny In Case Of Failure)

UCId	UC17.22
Use case	Client Based Content License Verification (Access Deny In Case Of Failure)

Description	This is a complementary step that has to be performed on the local device every time the content is accessed in order to ensure proper fruition and full respect of DRM rules and constraints
Actors	The local system, the end user fruition device
Assumptions	The device has enough computational power to manage at least a few security operation related to licensing check and operate consequently
Steps	1 The user tries access to a protected and governed AXMEDIS object 2 As the AXMEDIS object is governed, the domain PMS receives an authorisation request that includes the user identification, the right, the resource and optionally the license(s) or its(their) identifier(s). 3 The domain PMS obtains the licenses associated to the user from the database of DRM licenses. 4 The domain PMS performs the authorisation. 5 If the result of the authorisation is positive the user is authorised. If not, it returns the reasons why not. 6 Depending on results of the authorization proceeds as follows: 6.1 If the authorization has been positive, the object is unprotected and content fruition is allow 6.1.1 Decrements the iteration count 6.1.2 Informs customer of remaining iterations 6.2 If the authorization is negative, blocks content fruition 1.a.1 Informs customer of reasons why
Post-conditions	Depending on the content of the license (single/multiple fruition…) the licensing management module should proceed to allow/block further content fruition
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	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

17.23 Application Frontend Installation On End User Device

UCId	UC17.23
Use case	AXMEDIS viewer installation on end user device
Description	The end user is logged-on and using the kiosk to install the Application Frontend (including the AXMEDIS viewer).
Actors	The end user and the Kiosk Manager
Assumptions	The kiosk application is up and running, the back end is connected and functional.

Steps	1. The user selects in the Content List the option "Other Services" 2. The syspem presents the user a menu with the following options: a. Change profile b. Application Frontend install 3. The user selects on the kiosk GUI the option "Application Frontend install" 4. The system presents instructions for installation and how to connect wirelessly to the kiosk 5. The system asks the user to connect via web to a specific URL from his device 6. The user accesses the specified URL 7. The page loaded initiates the download of the application & viewer 8. The system informs the user of download results.
Post-conditions	The application front end is properly installed
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

17.24 User Profile Change

UCId	UC17.24
Use case	User Profile Change
Description	The end user is logged-on and willing to change own profile.
Actors	The end user and the Kiosk Manager
Assumptions	The kiosk application is up and running, the back end is connected and functional.

Steps	1 The application front end has granted access to available services including: 1.a Browse the catalogue 1.b Modify own data 1.c View support information 1.d Logout 2 The user selects in the Content List the option "Modify own data" 3 The system presents the user the profile form with the following data: • First Name [mandatory], • Last Name [mandatory], • Age [mandatory], • Address [mandatory]: • mail address, phone, mobile, e-mail, VAT code • … • Default Language [mandatory], • Preferred payment method: • pre-paid-cards, credit card • … • Payment method: • card #, validity from, validity to, type • … • Billing info: • mail address, phone, mobile, e-mail, VAT code • … • Preferred device: • PDA, • Smartphone, • Other 4 The user provides the required data 5 The user confirms input operation ending either pressing a button on the interface or any other widget.
Expected results	1. The user should be registered 2. The user should be assigned an AXMEDIS UID 3. The system should be notified of the registration (via mail/sms) 4. The user should be logged into the system
Post-conditions	The user profile is successfully updated
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

17.25 Interface Language selection

UCId	UC17.25
Use case	Interface Language Selection
Description	The user selects the interface language
Actors	The end user, the Kiosk Front end application
Assumptions	The kiosk front end application is fully functional.

Steps	1 The application front end exits from idle mode when a user interacts 2 The application front-end presents the user a screen with the flags of the supported languages for the GUI 3 The user presses the selected language 4 The application front end sets-up the environment variable stating the GUI language 5 The application front end presents the user a page for log-in / registration
Post-conditions	The kiosk application front end shall have adopted the selected language for the user interface
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None

17.26 User Device Configuration

UCId	UC17.26
Use case	User Device Configuration
Description	The end user configures own device (PDA…)
Actors	The end user, the Kiosk Front end application, the user fruition device (PDA…)
Assumptions	The kiosk front end application is fully functional.
Steps	1 The user has access to a page (either printed or in push) with the following info: 1.a How to connect the PDA / Tablet to the kiosk via WiFi (including how to test the connection) 1.b How to download the Application client on the device (including how to test the client) 2 The user performs on the device the required operation to configure the WiFi connection 3 The user performs the suggested check to ensure that WiFi configuration is successful 4 Device connects to the kiosk application front end 5 The application front end returns a test display page 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) 7 The device downloads the application client 8 The user install the downloaded client 9 The user performs the suggested check to ensure that application client install is successful 10 Device connects to the kiosk application front end 11 The application front end returns a test display object and a link to bookmark for future access via device 12 The application client displays the test object 13 The application client bookmarks the provided URL to access via device 14 The installed AXMEDIS client connects to the domain PMS to perform the requested “Registration” & “Authentication” as described in overall scenarios V3.9 (slide 219-220)
Post-conditions	The end user device is properly configured
Variations	None
Asynchronous actions	None
Design suggestions	None

Issues	None

17.27 Content Update (via Satellite)

UCId	UC17.27
Use case	Content Update (via Satellite)
Description	The kiosk content is updated via satellite
Actors	The kiosk manager, the Kiosk and the AXMEDIS framework
Assumptions	The kiosk and all its application are fully functional.
Steps	1 The checking time is over a Down-Link channel check has to be performed 2 The AXMEDIS B2B Satellite Reception Listener checks for data availability and behaves as follows: 2.a Data is not available yet so a further check is scheduled and the application enters wait mode (cycling back to point 1) 2.b Data is available therefore is downloaded (2.b.1) and progressively cached locally (2.b.2) 2.c Received data is stored locally 3 The AXMEDIS B2B Satellite Reception Listener activates the AXMEDIS Action Manager to decide how to proceed 4 The AXMEDIS Action Manager invokes the AXMEDIS B2B Satellite Reception Content Checker to verify consistency check on received data 5 The AXMEDIS B2B Satellite Reception Content Checker proceeds as follows 5.a Performs consistency check on received data 5.b If result is positive returns ACK and control to the AXMEDIS Action Manager 5.c If result is negative requires the distribution server to resend the damaged packages via Up-link as detailed here after: 5.c.1 Satellite Reception Content Checker requires missing or damaged packages via Up-Link 5.c.2 Satellite Reception Content Checker receives missing or damaged packages via Up-Link 5.c.3 Satellite Reception Content Checker returns ACK and control to the AXMEDIS Action Manager 6 The AXMEDIS Action Manager retrieves the data from the local storage 7 The AXMEDIS Action Manager extracts the content form the OpenSky package 8 The AXMEDIS Action Manager checks the received data to determine what it is and behaves consequently: 8.a Received data are AXMEDIS Object: data is stored in the AXDB 8.b Received data are system / application updates: invoke the kiosk data manager to store data locally according to needs 8.b.1 The kiosk data manager stores the received data locally in plain format
Post-conditions	The kiosk content is properly updated
Variations	None
Asynchronous actions	None
Design suggestions	None
Issues	None