Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE5.2.2.3
AXMEDIS Framework Validation and Integration Report, third update

Version: 1.0
Date: 28/04/2008
Responsible: Víctor Torres (UPC) (revised and approved by coordinator)

Project Number: IST-2-511299 Project Title: AXMEDIS Deliverable Type: report Visible to User Groups: yes Visible to Affiliated: yes Visible to the Public: yes
Deliverable Number: DE5.2.2.3 Contractual Date of Delivery: M42 Actual Date of Delivery: 24/04/2008 Title of Deliverable: AXMEDIS Framework validation and integration reporting Work-Package contributing to the Deliverable: WP5, WP4 Task contributing to the Deliverable: WP5, and WP4 Nature of the Deliverable: report Author(s): UPC, EXITECH, DSI

Abstract: This document contains the integration reports and the results of the test procedures for the AXMEDIS modules and applications integration.

Keyword List: validation verification

Table of Contents

1 INTRODUCTION

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 devoted to the description of tools validation and integration done inside WP5.2 and WP5.3.

Main deliverables in WP5 are:

- DE5.1.2.2 – AXMEDIS Framework for all, first update, M32
- DE5.0.1.2 – AXMEDIS Major Tools User Manuals and installation manual, first update, M34
- DE5.2.2.2 – AXMEDIS Framework Validation and Integration report, M36

WP5.2 -- Component Validation and Acceptance
This WP is coordinated by FUPF up to M24, then by UPC.
Period: M8-M48
In order to correctly set up the AXMEDIS framework, each implemented and supplied component should be validated and certified before allowing its use by content creators, content providers and final users inside AXMEDIS. The validation content is produced and collected in WP8 and the related test cases are defined in WP2 as described in WP5.1. The validation will be performed by skilled partners but different to those have created the module. Several industrial partners are involved in this work. To perform validation and acceptance of software components and tools, we have split this WP into several tasks (the great part of this work will be done after the first 18 months).

T5.2.1: Start up of the Component Validation and Acceptance
Managed by FUPF up to M24, then by UPC. The work should be assigned on the basis of competencies to all the partners involved on this WP.

- Definition of guidelines for deciding if a component is valid or not for its use in the AXMEDIS framework.
- Definition of acceptance guidelines for components.
- Specification of requirements for constructing tools and / or mechanisms for acceptance and validation. At this point, it should be decided which mechanisms for component validation and acceptance will be based on automatic tools and which ones will be manual.
- Development of tools to check the above guidelines.

T5.2.2: Periodic verification and Acceptance Testing

Managed by FUPF up to M24, then by UPC. The work should be assigned on the basis of competencies to all the partners involved on this WP.

- Software components and tools Testing of the tools developed to check the guidelines.
- Software components and tools Verification and validation of interoperability, etc.

The scope of the validation covers not only that of assessing the quality in terms of research value, but also that of verifying the: reliability, the robustness with respect to the test cases, and the conformance to the AXMEDIS framework.

This WP will create a feedback to the activity of research performed in WP4. Intended activities and targets of WP5.2 include:

- M11: Definition of validation and acceptance guidelines for deciding if a component can be accepted and integrated into the AXMEDIS framework.
- M12, M16, M20, M24, M28, M32, M36: Periodic validation and acceptance test.
- M16: Design and Development of tools to check the above guidelines.
- M20: Testing of the tools developed to check the guidelines.
- M30: Integration of tools in the AXMEDIS framework.
- M36: Verification and validation of component interoperability, etc.

WP5.3 -- AXMEDIS framework integration and maintenance
This WP is coordinated by EXITECH
Period: M9-M48
This WP is devoted to the integration of the components mentioned in the WP5.1. The integration work will permit the building of several demonstrators, for instance, the demonstrators developed in WP9 and the trials developed by the Take up actions. The great part of this work will be done after the first 18 months of project and will consist of:

T5.3.1: set up of the AXMEDIS framework for integration
Managed by EXITECH

- Components data base set up and data base management.
- CVS set up and management for source sharing.

T5.3.2: Continuous integration of AXMEDIS components
Managed by EXITECH, performed by all partners involved according to their skill and related tools of which they are responsible.
Activity of integrating into the CVS the components, resolving conflicts, supporting the other partners during the integration. This will lead to refine the guidelines and also the stubs of the modules.

T5.3.3: Regression and integration testing
Managed by EPFL up to M22 then by FUPF up to M24, then by UPC, performed by all partners involved according to their skill and related tools of which they are responsible.
Regression and integration test of the software components to verify their global functionalities, by using the integration test cases defined in WP2 and the data set produced and collected in WP8.

T5.3.4: Optimisation of AXMEDIS components
Managed by DSI, performed by all partners involved according to their skill and related tools of which they are responsible.
Optimisation of AXMEDIS Components for improving the quality of their results and removing potential integration problems.

This WP will create a feedback to the activity of research performed in WP4. Intended activities and targets of WP5.3 include:

- M13: AXMEDIS Framework integration guidelines, CVS rules, access to AXMEDIS framework database. Start up of the CVS for the AXMEDIS framework.
- M16, M22, M28, M34, M40, M48: Continuous integration and refinement of AXMEDIS components: regressing testing, optimisation, etc.
- M18: Integration of DRM components into the framework.
- M24: Testing of the component integration to verify their global functionalities by using the test cases defined in WP2 and the data set produced and collected in WP8.
- M30: Optimisation of the components for improving the quality of their results and removing the eventual integration problems (with participation of all partners providing tools).
- M36: Test of the optimised components.

2 AXMEDIS COMPONENT VALIDATION AND ACCEPTANCE (UPC)

2.1 Guidelines for component Validation and Acceptance

In order to correctly set up the AXMEDIS framework, each implemented and supplied component should be validated and certified before allowing its use by content creators, content providers and final users inside AXMEDIS. This will help in guaranteeing that the implemented component meets the requirements.

The validation will depend on several issues and will seek to establish the degree to which a given component fulfils the following performance criteria:

- It does what it is supposed to do
- It does its job in a reasonable time
- It can be integrated with other components
- It is reliable
- It is robust
- It accomplishes test cases
- It is conformant with the AXMEDIS framework

In the following sections we will describe the general guidelines for component validation and acceptance in the AXMEDIS framework.

2.1.1 Component validation

The component validation process should involve the checking and analysis of the component in order to verify that the component meets the requirements demanded of it.

The main validation activities are the revision and testing of the component.

Review phase main steps:

- Review phase involves the manual review of the component, directly evaluating it. It will help in determining that the requirements, design concepts and specifications have been met. The revision of a component can include several activities, like peer reviewing or code inspection.
- The result of the review should be a report where the reviewing team explains the revision performed, the errors found and other.

Testing phase involves the testing of the component at different levels, from unit test through to system test.

The main steps in the testing phase are the following ones:

- During development phase of the component, unit tests should be done in order to check that the functionality being implemented is the correct one.
    - This task should be done by the development team of the component, as they have the complete knowledge over it.
        - Modifications over the specification should be reported

- When the development of a component has been completed, an initial unit testing phase and an integration test is needed to evaluate how the unit performs and how it interacts with other units.
    - This task should be done with the cooperation of the development teams of the integrated components. The result of this task should be reported in order to perform the needed actions for solving the problems found.
    - This task also needs to evaluate the following aspects, and report on them:
        - Documentation provided
        - Differences with the specification, if any
        - Interfaces among components

- The rest of tests should involve the whole system and it will be different depending on the demonstrator. So system testing should be demonstrator testing for the AXMEDIS framework, at least in a first phase.

In order to perform the above tests, the test cases identified and described in WP2 and the content for test cases created and described in WP8 should be taken into account.

Apart from tests and reviews, component validation should also include the examination of the degree to which the component is properly documented, including updates in the specification documents, documentation of the programming interface, usage manual (specially for end-user or business-user applications) or installation manual. This information should be included in the AXMEDIS framework part associated to this component.

The major requirements and functionalities of the component need to be mapped to the component under validation, in order to check that the component accomplishes them.

2.1.2 Component Submission

This section describes what has to be reported when a component is submitted into the AXMEDIS framework.

Module name	Name of the module being reviewed.
Module description	General purpose of the module.
Major requirements	To be provided by the component owner
Major use cases	To be provided by the component owner
Major related components	To be provided by the component owner
List of Test Cases	• To be provided by the component owner • Accepted by the validators Additional test cases may be provided.
Used Libraries	To be provided by the component owner Versions and library related information should be provided
Languages	Programming languages
Operating system(s)	A list and information to compile for different OS
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

2.1.3 Review report form

This section gives an example of what has to be reported when a module is being reviewed. The main information to be reported is described in the following table.

Review ID	Identifier of the review.
Module names	Name of the modules being reviewed.
Module description	General purpose of the module
List of use cases	Related Use Cases
List of Test Cases	Related Test Cases
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.
Participants	Names, organisations and roles of people involved in the review.

ID	**Who**	**Date**	**Issue location and description**
Issue number	Name and organisation of the person who finds the	Date when the issue is found	Part of the module where the issue is found and description of the issues found in the module during revision and. The location could be the source code file, web page, etc; it will depend on the type of application.

	issue		

2.1.4 Verification report form

This section gives an example of what has to be reported when a module is verified, after it has been reviewed and some issues have been found. The information to be reported is described in the following table.

Verification ID	Identifier of the verification.
Review ID	Identifier of the review to which this verification refers.
Module names	Name of the modules being verified.
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

ID	**Who**	**Date**	**Status**	**Issue description and resolution**
Related Review Issue number	Name and organisation of the person who verifies the issue	Date when the verification is performed	Solved, Proposed or Not Solved	How the issue found during the revision process has to be solved or has been solved

Regarding the Status column, it has to be filled in the following manner:

- Solved: when the issue in the review report has been solved
- Proposed: when the issue in the review report has been proposed but not solved
- Not solved: when the issue in the review report has been neither proposed nor solved

2.1.5 Component acceptance

Component acceptance mostly involves those components which are addressed to application users, either business or final users.

Acceptance testing allows users (or project partners which can assume this role), to test the functionality of the system against the requirements and use cases. Each kind of tool should be tested by a key user on this area. For instance, the component acceptance test of a content production tool should be done by a user that is skilled in the area together with part of the development team, in order to get the comments on the tool behaviour.

It will be very useful to follow the test cases that involve components to be accepted in order to check if they are correct. A report on if test cases are accomplished or not should be done by test participants.

2.1.6 Start up of component validation and acceptance

In order to start up the component validation and acceptance in the AXMEDIS framework, the responsible of each module should:

- Follow the CVS repository guidelines
- Prepare unit tests
- Perform unit tests and give a brief report on them
- Prepare integration tests
- Perform integration tests. They may be done by the partners participating in the integration together or not. In the latter case, some communication mechanisms will be established in order to solve possible integration problems as soon as possible. For instance, if one partner is performing an integration test of one of his components with other partner's component, a videoconference or audio-conference might be set up before, during or after tests are done

- Give a brief report on the integration test, outlining problems found, categorising them (application error, bad parameter value, inconsistent API, etc)

These reports should be available in the portal to the rest of the partners in order to improve the component knowledge and maintenance. A new directory / portal section should be created to contain them. They could also be included into the CVS repository, together with the corresponding application, web service, library, etc.

2.1.7 Periodic verification

During project development, components will be improved / updated to solve problems found during the different testing phases or to meet new requirements found during the acceptance tests or the evolution of commercial software products and standards affecting the AXMEDIS framework components.

In this way, there will be the need to inform the rest of the partners of any components having been updated, so that new integration tests are performed if needed.

Reports on the unit tests performed over the new or updated components have to be given together with the new version of the component.

The periodic verification steps should be as follows:

1 Update component into the corresponding directory of the CVS repository, indicating that it is a new version.
2 Send e-mail to the reflector and / or developers mailing list to inform that an update on a component has been done and including the results of the related regression test(s) when needed.
3 Partners using the updated component should:
  3.1 Perform integration tests with the new version of the component. The participation / support of the component responsible may be requested.
  3.2 Report errors / problems detected during the integration test, if any. The report has to be uploaded in the portal, in order that partners are informed of the results of the test.
  3.3 If needed, after integration errors / problems reported in 3.2 are solved, components using the initially updated component should be also updated in the CVS repository.
  3.4 Go to verification step 2 for the component(s) updated in step 3.3.

As a modification in one component may involve the modification of many other components, regression and integration tests should be systematically done. In some cases, it could happen that a component will no longer work with lower versions of libraries and components. This should be specified in the documentation of the component.

2.1.8 Acceptance testing

Acceptance of a component involves that it has previously been validated and verified, making the corresponding unit and integration tests. Once they have been passed, acceptance tests should be done. They may involve other partners, final users, user group experts, etc.

Acceptance test results could be reported using the review form described in section 2.1.3.

Once the component has been accepted, it can be made publicly available in the AXMEDIS framework. Moreover, it may be further optimised.

3 AXMEDIS FRAMEWORK VALIDATION (EXITECH)

3.1 Introduction

In this section an overview of the formal validation performed on the AXMEDIS VCS is reported. Formal validation consists in the verification that all the requested items for the VCS are present in the VCS for each part and for each partner. An overview in terms of overall values and per partner evaluation is also reported.
The overall evaluation of the validation is:

- 0 or OK, that means that all is OK;
- 1 or VER, that means that some small activities have to be done in order to reach the upper level. The situation can be recovered very easily;
- 2 or ER, that means that a medium effort has to be done in order to recover the situation. In general some documentation or library is missing
- 3 or TBR, that means that the situation has to be recovered since the object is not acceptable and usable in its current shape.

3.2 Applications formal validation

3.2.1 General evaluation per partner

In this section the evaluation of partner involved in the Application section is described and summarized in the following table:

#components	partner	OK	VER	ER	TBR	OVERALL
2	**ILABS**	100%	0%	0%	0%	0
3	**UNILEEDS**	100%	0%	0%	0%	0
9	**EXITECH**	56%	0%	0%	44%	1,333333333
7	**FUPF**	29%	0%	0%	71%	2,142857143
2	**SEJER**	0%	0%	50%	50%	2,5
11	**DSI**	0%	0%	36%	64%	2,636363636
3	**HEXAGLOBE**	0%	0%	0%	100%	3
3	**TISCALI**	0%	0%	0%	100%	3
1	**EPFL**	0%	0%	0%	100%	3
2	**UR**	0%	0%	0%	100%	3
1	**KTU**	0%	0%	0%	100%	3

It is evident that UNILEEDS and ILABS have provided all is needed according to the repository requirements, EXITECH can improve, while ALL the other partners need a large improvement in performances.

3.2.2 General evaluation of the Application section

In this section the general evaluation of the Application section component by component is reported below and is depicted in the following figure.

	APPLICATIONS - v1977 - newrepos		FOLDERS										
	needed:						howToBuild		howToDeploy				howToTest
	optionals:		mand.4 C/C++					yes		yes			
		NOTES:	source	bin	lib	include	project	doc specificatio	doc configuration-deployme	doc code	doc other	doc test	
DSI	axactivex	3	cpp					empty		empty			
DSI	axcpquickstarter/	3	cpp					empty		empty			
EXITECH	AXDBBrowser	0											
DSI	axeditor	2	cpp				howToBuild	empty					
HEXAGLOBE	axeptool	3 index.txt missing	STRANGE FOLDERS HERE										
UR	AxmedisConstraintEngine/	3	c	STRANGE FOLDERS HERE									
SEJER	AxMozillaPlayer	3				empty	howToBuild						
SEJER	axmozillaplugin	2	cpp				howToBuild	empty					
DSI	axplayer	3 index.txt missing	cpp					empty		empty			
UNILEEDS	axpnpeng	0	cpp										
TISCALI	axPocketPCPlayer/	3	STRANGE FOLDERS HERE										
DSI	axskinplayer	3	cpp				howToBuild						
KTU	axstbplayer/	3	cpp				howToBuild						
HEXAGLOBE	azureus_library	3 index.txt missing	STRABGE FOLDERS HERE										
UR	BizTalkWorkflow/	3	STRANGE FOLDERS HERE										
HEXAGLOBE	bt_tracker	3 index.txt missing	STRABGE FOLDERS HERE										
DSI	collector indexer	2 index.txt missing	cpp									howToTest	
FUPF	contractanalyser	3	cpp					empty				empty	
FUPF	contractmanager	3											
FUPF	contractgen	0	cpp										
FUPF	conversionassistant	3								empty			
FUPF	domainmanager	3	cpp		empty		howToBuild						
FUPF	drmeditorviewer (ex drmeditor)	0	cpp										
FUPF	DUDManager/	3	cpp										
EXITECH	ExitechWSClient	0											
ILABS	kioskcomponents	0											
TISCALI	mediaclub	3 index.txt missing	STRANGE FOLDERS HERE										
UNILEEDS	metadatamapperGUI	0	cpp										
ILABS	mobilecomponents	0 index.txt missing											
DSI	p2ppublishing	3	STRANGE FOLDERS HERE										
EXITECH	P2PQueryServerUploader	0											
UNILEEDS	pnpeditor	0	cpp										
EXITECH	QueryWebUserInterface	0											
EXITECH	RegistrationPortal	0											
DSI	ruleeditor	2					howToBuild	empty					
DSI	ruleexecutor	3 wrong folder "client" and stange folders in the root	cpp			empty	howToBuild	empty		empty			
DSI	rulescheduler	2 strange clipart folder	cpp				howToBuild	empty					
DSI	SIAE-COPOP/	3	STRANGE FOLDERS HERE										
EXITECH	SIAE-DistributionPortal/	3	STRANGE FOLDERS HERE										
EXITECH	SIAE-PaymentPortal/	3	STRANGE FOLDERS HERE										
TISCALI	smil_mobile_player	3 strange clipart folder	STRANGE FOLDERS HERE										
EPFL	smil_to_xmto	3 strange clipart folder	cpp				howToBuild						
EXITECH	Streaming_mobile/	0											
EXITECH	Streaming_PC/	3	STRANGE FOLDERS HERE										

The major part of the RED blocks are in the documentation area that means that no documentation or strongly incomplete documentation has been provided for the component.
This can foresee future problems if someone will have to take in charge components without enough documentation.

3.2.3 Overall evaluation

The overall figure that is evaluated as a weighted sum over all the components for application is not good being 2 over 3, where 0 is the best and 3 the worst.

3.3 Framework formal validation

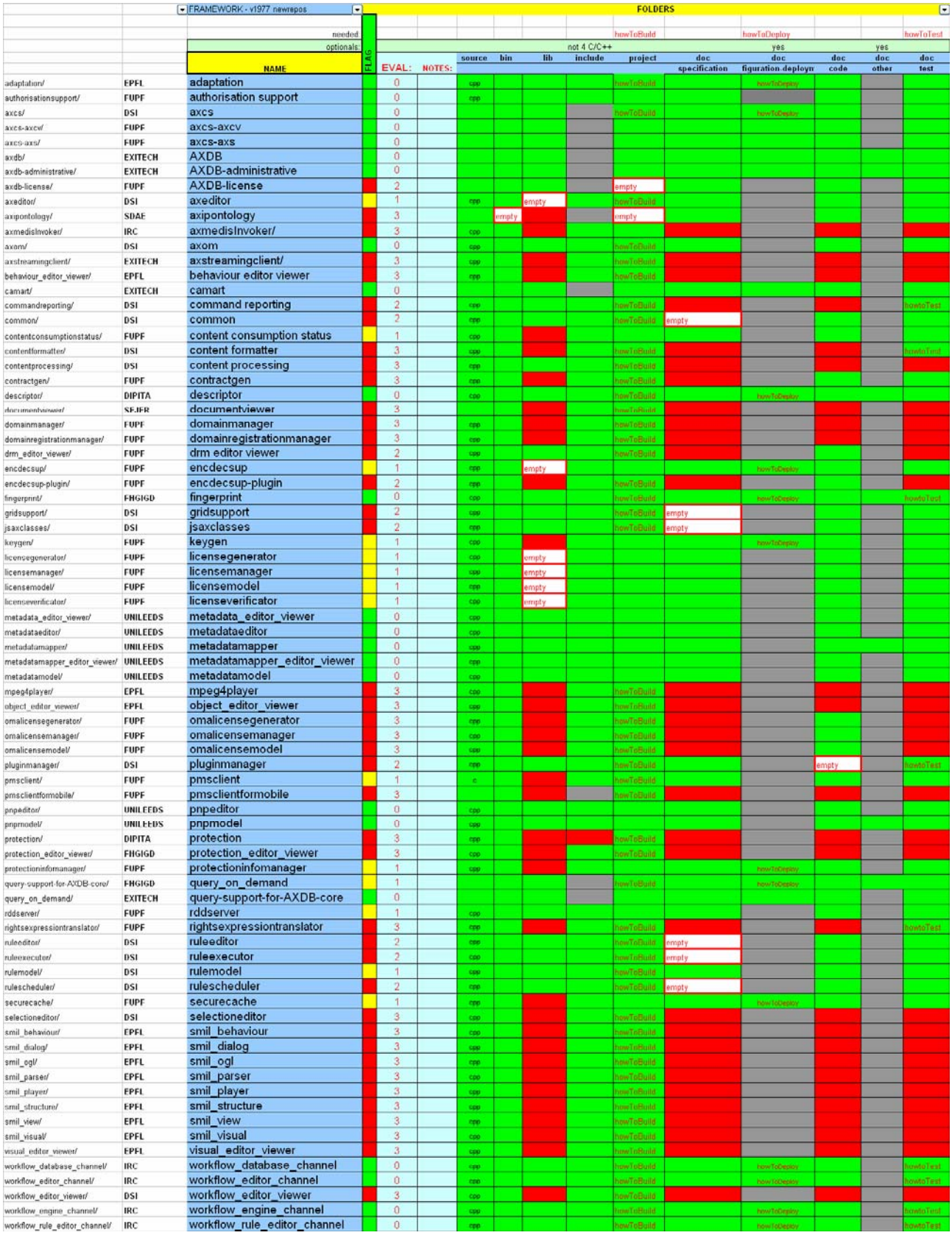
3.3.1 General evaluation per partner

The following figure evidences the partners that have provided complete information and code.
The partners that have performed a good work are UNILEEDS, EXITECH and IRC, while FUPF (UPC), DSI and EPFL having in charge more that 70% of the framework need a large improvement and SDAE and SEJER needs improvement also if the effect will be relatively small since they cover only less than 3% of the framework.

			OK	VER	ER	TBR	OVERALL
8,97%	7	UNILEEDS	100,00%	0,00%	0,00%	0,00%	0,00
6,41%	5	EXITECH	80,00%	0,00%	0,00%	20,00%	0,60
6,41%	5	IRC	80,00%	0,00%	0,00%	20,00%	0,60
3,85%	3	FHGIGD	33,33%	33,33%	0,00%	33,33%	1,33
2,56%	2	DIPITA	50,00%	0,00%	0,00%	50,00%	1,50
32,05%	25	FUPF	8,00%	44,00%	12,00%	36,00%	1,76
20,51%	16	DSI	12,50%	12,50%	50,00%	25,00%	1,88
16,67%	13	EPFL	7,69%	0,00%	0,00%	92,31%	2,77
1,28%	1	SDAE	0,00%	0,00%	0,00%	100,00%	3,00
1,28%	1	SEJER	0,00%	0,00%	0,00%	100,00%	3,00

3.3.2 General evaluation of the section

In the following picture a general overview over the Framework is given and as evidenced also for application the main problems are in the documentation area where large block of RED are present.
Conclusion are therefore similar to Application section.



		FRAMEWORK - v1977 newrepos			FOLDERS										
		needed							howToBuild		howToDeploy			howToTest	
		optionals			not 4 C/C++						yes		yes		
		NAME	FLAG	EVAL: NOTES:	source	bin	lib	include	project	doc specification	doc figuration deploym	doc code	doc other	doc test	
adaptation/	EPFL	adaptation		0	cpp				howToBuild		howToDeploy				
authorisationsupport/	FUPF	authorisation support		0	cpp										
axcs/	DSI	axcs		0					howToBuild		howToDeploy				
axcs-axcv/	FUPF	axcs-axcv		0											
axcs-axs/	FUPF	axcs-axs		0											
axdb/	EXITECH	AXDB		0											
axdb-administrative/	EXITECH	AXDB-administrative		0											
axdb-license/	FUPF	AXDB-license		2					empty						
axeditor/	DSI	axeditor		1	cpp		empty		howToBuild						
axipontology/	SDAE	axipontology		3		empty			empty						
axmedisInvoker/	IRC	axmedisInvoker/		3	cpp										
axom/	DSI	axom		0	cpp				howToBuild						
axstreamingclient/	EXITECH	axstreamingclient/		3	cpp				howToBuild						
behaviour_editor_viewer/	EPFL	behaviour editor viewer		3	cpp				howToBuild						
camart/	EXITECH	camart		0											
commandreporting/	DSI	command reporting		2	cpp				howToBuild					howtoTest	
common/	DSI	common		2	cpp				howToBuild	empty					
contentconsumptionstatus/	FUPF	content consumption status		1	cpp										
contentformatter/	DSI	content formatter		3	cpp				howToBuild					howtoTest	
contentprocessing/	DSI	content processing		3	cpp				howToBuild						
contractgen/	FUPF	contractgen		3	cpp				howToBuild						
descriptor/	DIPITA	descriptor		0	cpp				howToBuild		howToDeploy				
documentviewer/	SEJER	documentviewer		3					howToBuild						
domainmanager/	FUPF	domainmanager		3	cpp				howToBuild						
domainregistrationmanager/	FUPF	domainregistrationmanager		3	cpp				howToBuild						
drm_editor_viewer/	FUPF	drm editor viewer		2	cpp				howToBuild						
encdecsup/	FUPF	encdecsup		1	cpp		empty				howToDeploy				
encdecsup-plugin/	FUPF	encdecsup-plugin		2	cpp				howToBuild						
fingerprint/	FHGIGD	fingerprint		0	cpp				howToBuild		howToDeploy			howtoTest	
gridsupport/	DSI	gridsupport		2	cpp				howToBuild	empty					
jsaxclasses/	DSI	jsaxclasses		2	cpp				howToBuild	empty					
keygen/	FUPF	keygen		1	cpp						howToDeploy				
licensegenerator/	FUPF	licensegenerator		1	cpp		empty								
licensemanager/	FUPF	licensemanager		1	cpp		empty								
licensemodel/	FUPF	licensemodel		1	cpp		empty								
licenseverificator/	FUPF	licenseverificator		1	cpp		empty								
metadata_editor_viewer/	UNILEEDS	metadata_editor_viewer		0	cpp										
metadataeditor/	UNILEEDS	metadataeditor		0	cpp										
metadatamapper/	UNILEEDS	metadatamapper		0	cpp										
metadatamapper_editor_viewer/	UNILEEDS	metadatamapper_editor_viewer		0	cpp										
metadatamodel/	UNILEEDS	metadatamodel		0	cpp										
mpeg4player/	EPFL	mpeg4player		3	cpp				howToBuild						
object_editor_viewer/	EPFL	object_editor_viewer		3	cpp				howToBuild						
omalicensegenerator/	FUPF	omalicensegenerator		3	cpp				howToBuild						
omalicensemanager/	FUPF	omalicensemanager		3	cpp				howToBuild						
omalicensemodel/	FUPF	omalicensemodel		3	cpp				howToBuild						
pluginmanager/	DSI	pluginmanager		2	cpp				howToBuild			empty		howtoTest	
pmsclient/	FUPF	pmsclient		1	c				howToBuild						
pmsclientformobile/	FUPF	pmsclientformobile		3					howToBuild						
pnpeditor/	UNILEEDS	pnpeditor		0	cpp										
pnpmodel/	UNILEEDS	pnpmodel		0	cpp										
protection/	DIPITA	protection		3	cpp				howToBuild						
protection_editor_viewer/	FHGIGD	protection_editor_viewer		3	cpp				howToBuild						
protectioninfomanager/	FUPF	protectioninfomanager		1	cpp						howToDeploy				
query-support-for-AXDB-core/	FHGIGD	query_on_demand		1					howToBuild		howToDeploy				
query_on_demand/	EXITECH	query-support-for-AXDB-core		0											
rddserver/	FUPF	rddserver		1	cpp										
rightsexpressiontranslator/	FUPF	rightsexpressiontranslator		3	cpp				howToBuild					howtoTest	
ruleeditor/	DSI	ruleeditor		2	cpp				howToBuild	empty					
ruleexecutor/	DSI	ruleexecutor		2	cpp				howToBuild	empty					
rulemodel/	DSI	rulemodel		1	cpp				howToBuild						
rulescheduler/	DSI	rulescheduler		2	cpp				howToBuild	empty					
securecache/	FUPF	securecache		1	cpp						howToDeploy				
selectioneditor/	DSI	selectioneditor		3	cpp				howToBuild						
smil_behaviour/	EPFL	smil_behaviour		3	cpp				howToBuild						
smil_dialog/	EPFL	smil_dialog		3	cpp				howToBuild						
smil_ogl/	EPFL	smil_ogl		3	cpp				howToBuild						
smil_parser/	EPFL	smil_parser		3	cpp				howToBuild						
smil_player/	EPFL	smil_player		3	cpp				howToBuild						
smil_structure/	EPFL	smil_structure		3	cpp				howToBuild						
smil_view/	EPFL	smil_view		3	cpp				howToBuild						
smil_visual/	EPFL	smil_visual		3	cpp				howToBuild						
visual_editor_viewer/	EPFL	visual_editor_viewer		3	cpp				howToBuild						
workflow_database_channel/	IRC	workflow_database_channel		0	cpp				howToBuild		howToDeploy			howtoTest	
workflow_editor_channel/	IRC	workflow_editor_channel		0	cpp				howToBuild		howToDeploy			howtoTest	
workflow_editor_viewer/	DSI	workflow_editor_viewer		3	cpp				howToBuild						
workflow_engine_channel/	IRC	workflow_engine_channel		0	cpp				howToBuild		howToDeploy			howtoTest	
workflow_rule_editor_channel/	IRC	workflow_rule_editor_channel		0	cpp				howToBuild		howToDeploy			howtoTest	

3.3.3 Overall evaluation

The overall evaluation of Framework is only slightly better with respect to Application being 1.61 over 3, where 0 is the best and 3 the worst.

This figure needs a large improvement since the framework is the core part of AXMEDIS and contains also a huge number of components and subcomponents.

3.4 Web services formal validation

3.4.1 General evaluation per partner

Partners operating over the Framework section show a better performance since only DSI and SDAE needs a small improvement.

#components		OK	VER	ER	TBR	OVERALL
8	EXITECH	100%	0%	0%	0%	0,00
9	FUPF	33%	22%	11%	33%	0,89
6	DSI	67%	0%	0%	33%	1,00
1	SDAE	0%	100%	0%	0%	1,00

3.4.2 General evaluation of the section

The overview of the section shows a large green area that implies a good shape of this section.

	WEBSERVICES - v1977 NEWREPOS		FOLDERS										
	needed:						howToBuild		howToDeploy				howToTest
	optionals:		not 4 C/C++						yes			yes	
			source	bin	lib	include	project	doc specification	doc configuration-deployme	doc code	doc other	doc test
FUPF	AXCSAXCVWs	0										
FUPF	AXCSAXSWs	0										
DSI	AXCSObjectRegistrator	0										
DSI	AXCSReporting	0										
DSI	AXCSStatistics	0										
DSI	AXCSUserRegistrator	0										
EXITECH	AXDBDelWS	0										
EXITECH	AxdbQSWs	0										
SDAE	AXIPOntologyWS	2					howToBuild					
EXITECH	AXDBSupportWS	0										
EXITECH	LoadeSaverWs	0										
EXITECH	LockUnlockWs	0										
FUPF	OMALicenseCreatorWS	3	cpp				howToBuild					
EXITECH	P2PQuerySupportWs	0										
FUPF	PARManager	1	cpp		empty		howToBuild					
FUPF	ParQuerySupport	2					howToBuild		howToDeploy	empty		empty
FUPF	PMSDomainFactory	1										empty
FUPF	PMSDomainHome	1										empty
FUPF	PMSWs	0										
FUPF	PMSWsMT	0										
EXITECH	QuerySupportWs	0										
DSI	SIAE-AxcpNotificationWS	3										
DSI	SIAE-UploadServlets	3										
EXITECH	UserSelection Archive	0										

3.4.3 Detailed report

The overall evaluation of this section is very good: 0.67 over 3 where 0 is the best.

4 AXMEDIS FRAMEWORK REGRESSION TESTING (UPC)

4.1 Introduction

4.1.1 Definition

Regression testing is any type of software testing which seeks to uncover regression bugs. Regression bugs occur whenever software functionality that previously worked as desired stops working or no longer works in the same way that was previously planned. Typically regression bugs occur as an unintended consequence of program changes.

Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have reemerged.

Experience has shown that as software is developed, this kind of reemergence of faults is quite common. Sometimes it occurs because a fix gets lost through poor revision control practices (or simple human error in revision control), but just as often a fix for a problem will be "fragile" - i.e. if some other change is made to the program, the fix no longer works. Finally, it has often been the case that when some feature is redesigned, the same mistakes will be made in the redesign that were made in the original implementation of the feature.

Therefore, in most software development situations it is considered good practice that when a bug is located and fixed, a test that exposes the bug is recorded and regularly retested after subsequent changes to the program. Although this may be done through manual testing procedures using programming techniques, it is often done using automated testing tools. Such a 'test suite' contains software tools that allows the testing environment to execute all the regression test cases automatically; some projects even set up automated systems to automatically re-run all regression tests at specified intervals and report any regressions. Common strategies are to run such a system after every successful compile (for small projects), every night, or once a week.

Regression testing is an integral part of the extreme programming software development methodology. In this methodology, design documents are replaced by extensive, repeatable, and automated testing of the entire software package at every stage in the software development cycle.

Source: http://en.wikipedia.org/wiki/Regression_testing

4.1.2 Free tools

The most common way for regression testing is to use unit tests. Generally unit test tools are language relatives. There are a lot of free tools available for regression testing in the languages used in AXMEDIS. These are some free popular unit testing tools for C++, C and Java:

http://cppunit.sourceforge.net/
http://unitpp.sourceforge.net/
http://check.sourceforge.net/
http://junit.sourceforge.net/

4.1.3 Regression testing in AXMEDIS

Due to the fact that most implementations of Axmedis are in C++, CppUnit seems to be the best choice. CppUnit is one of the most popular C++ testing free tools and because it is originally a port of JUnit, it will be close to Java (for Java features).

The guidelines for applying Regression Testing in AXMEDIS are the following:

- Every unit of the whole project which has predictable results or results that can be automatically tested should be tested.

- It is in charge of every partner to make his own tests.
- Regression test have to run with the actual version of the unit. Or the actual results (tests passed/failed/error) must be available with the regression tests.
- A modification of a unit with regression tests shouldn't be submitted without executing the regression tests. The result must be as good as the previous results (with no result specified, all tests must pass). If better results are obtained with a modification, don't forget to post the new results expected with it.

One important thing to remember is that these kinds of tests are logical tests and consequently not user interfaces tests.

4.2 Testing the AXMEDIS framework modules

In the AXMEDIS repository, there is a specific folder for each of the modules devised to contain all the details regarding the tools that have been specifically developed and necessary steps for testing the single module or even its integration with external modules. In this sense a HowToTest.txt file is provided with the instructions, together with the needed executables, programming projects and related files if necessary.

An example of regression testing is that performed for AXMEDIS Certification and Verification module. In this case, JUnit has been used to test AXCV library as well as AXCV Web service and their integration with AXCS database and AXMEDIS Certification Authority. Testing tool description is included in the documentation associated to AXMEDIS Certification and Verification module. The Java source code and classes are provided together with a .bat file that executes the Java classes automatically on a Windows environment. In this way, the JUnit-based testing tool can be easily executed after introducing any changes to the module to see if the results still are the expected. If some important interfaces or functionality is changed, then the JUnit classes may need to be changed and recompiled.

Another example of regression testing is that performed for the AXMEDIS Database and Query Support modules. In this case Unit tests are adopted for testing the layer in a white box manner. The Unit test of the higher level is used to test in a black box manner the underlying layer. A functional test to check in black box manner the last layer will be implemented if needed. The testing is completely automated by using JUNIT or the ANT scripts provided with the source code in the AXMEDIS repository.

In other cases, specific testing applications are provided. It is the case e.g. of Secure Cache module, where a specific C++ project and executable is provided for performing some predefined tests over the module.

Finally, there are some other modules which are difficult to be tested on their own, but which can be easily tested when using the applications that involve them. It is the case of e.g. axeditor, ruleeditor, ruleexecutor and rulescheduler. In this case a reference is made to the application that uses them, which can be tested by the means described in its own HowToTest.txt file and testing folder.

5 AXMEDIS FRAMEWORK INTEGRATION AND MAINTENANCE (UPC)

5.1 Introduction

This section aims to describe how to validate the AXMEDIS Framework by providing the tools or procedures to perform an integration testing of the different modules.

For this purpose, it is assumed that a Unit and Regression testing initial step has been already performed on the AXMEDIS modules, so that what has to be tested here is not the correctness of the modules operation but the interaction and compatibility with other modules.

Section 5.2 describes the general procedure was initially proposed to be followed for all the AXMEDIS modules, that is, a general description of other modules with which the module integrates, a description o how integration testing needs to be done and a standard means for reporting the errors found during a review plus the reports used to describe the proposed or implemented solutions.

Section 5.3 summarizes the main topics that have arised during the integration of different AXMEDIS modules, applications and services.

5.2 General procedure for testing Integration in AXMEDIS

For each of the AXMEDIS modules the following information should be provided.

5.2.1 Modules with which it integrates

A list of modules that interact directly or indirectly with the present module and against which some aspects have to be verified to have a successful integration of the AXMEDIS Framework.

5.2.2 How to perform Integration Testing

A reference to the tools that have been created for the automatic or semi-automatic testing of the integration with other modules or, when automatic tests become difficult to be developed, the description of the procedure to be followed and the aspects to be verified during the integration testing.

5.2.3 Review report form

The reports regarding the results obtained during the review of the integration of different modules. Each report describes the issues and errors found during the review.

The main information to be reported when a module is being reviewed is described in the following table.

Review ID	Identifier of the review.
Module names	Name of the modules being reviewed.
Integration description	General purpose of the integration test review.
List of use cases	Related Use Cases
List of Test Cases	Related Test Cases
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.
Participants	Names, organisations and roles of people involved in the review.

ID	Who	Date	Issue location and description
Issue number	Name and organisation of the person who finds the issue	Date when the issue is found	Part of the module where the issue is found and description of the issues found in the module during revision and. The location could be the source code file, web page, etc; it will depend on the type of application.

5.2.4 Verification report form

Verification reports refer to the resolution or resolution proposal regarding review reports. They have to be created when a module is verified, after it has been reviewed and some issues have been found.

The information to be reported is described in the following table.

Verification ID	Identifier of the verification.
Review ID	Identifier of the review to which this verification refers.
Module names	Name of the modules being verified.
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

ID	Who	Date	Status	Issue description and resolution
Related Review Issue number	Name and organisation of the person who verifies the issue	Date when the verification is performed	Solved, Proposed or Not Solved	How the issue found during the revision process has to be solved or has been solved

Regarding the Status column, it has to be filled in the following manner:

- Solved: when the issue in the review report has been solved
- Proposed: when the issue in the review report has been proposed but not solved
- Not solved: when the issue in the review report has been neither proposed nor solved

5.3 Integration testing in AXMEDIS

Deliverable DE 5.2.2 described the modules with which each of the AXMEDIS modules integrates and the methodology for performing integration tests. Mainly, the procedure for performing integration tests consists on following the steps described in DE 2.2.1.2, the Test Cases Document, which can involve an appropriate initialisation of the related databases or other modules. Some of these steps can be performed in an automatic manner, by means of a specific executable, but some others involve the usage of Graphical User Interfaces, which depend on user performing them.

During this period, Review reports and Verification Reports have proved not to be very flexible, as the interaction with related partners becomes quite slow due to the overload of having to maintain a permanently updated report document. For this purpose, a reporting tool was developed. It enables the inclusion of review and verification reports in a semi-automatic manner and obtaining them in .doc predefined format when needed. The reporting tool can be accessed at http://cameron.upf.es/axmedis, using “demo” as user and password.

However, a direct interaction between the partners by means of email and messaging programs such as MSN, Skype, etc. has ended to be better for obtaining quick response from the involved partners. When any error has been detected or solved an email to the developers and/or general mailing list has been sent to inform about the topics.

Another useful mechanism fo testing the integration has been the generation of the AXMEDIS Tools (AXTools). The AXTools consist of an installshield which includes all major AXMEDIS Tools that can be freely distributed for being installed and used by AXMEDIS partners as well as non-partners. The production of the AXMEDIS Tools has become an important means for proving the integration of the modules in the AXMEDIS Framework as well as the interaction of different AXMEDIS applications.

5.4 Integration testing results

In this section we are going to summarize the main problems that have been detected when performing integration tests for the AXMEDIS Framework.

5.4.1 AXCP Tools

Issue 1

Experimenting the AXMEDIS tools, we found that among other content processing tools the Axmedis Editor also include an audio/video fingerprinting extractor. We started experimenting with it, although this activity is not related in any way to what we are doing in AXMEDIS/AX4HOME - we are just interested to test the performances of the tool.
Unfortunately we found a problem in the fingerprinting extraction that causes the AXMEDIS Editor to fail, and to continue to fail even after reinstalling all AXMEDIS tools!

Scenario:
An axmedis object with three windows media video files added to it. The files sizes are: 45Mb, 33Mb, and 0.16Mb. The user is utilizing the audio fingerprinting plugin to calculate (or compare) audio fingerprintings.
Description of user utilization:
The user initiates the fingerprinting calculation by pressing the execute button in the plugin window. Then, if the user tries to move the window during the fingerprinting computation, the mouse icon changes to the sand clock icon and Axmedis main window icon (in the upper-left side), changes to the "not responding" type icon. The program seems to be unstable, in this situation the fingerprinting calculation can continue and eventually finish, but if the user tries to close the main window the program hangs up.
Errors reported after the above described behavior:
1. Windows popup error message stating "ERROR: Unknown Exception". This is shown during the start-up of Axmedis editor. After pressing the OK button the editor continues its initialization and it starts running normally.
2. Windows popup error message stating "ERROR: CONTENT PROCESSING HAS NOT BEEN INITIALIZED". This message appears when the user tries to open the "Content processing plugins" window by clicking the right mouse button over the video resource icon.

Solution to Issue 1
Solution not yet provided by DSI

Issue 2
While using the Rule Editor included in the current released version of the AXMEDIS tools (axmedis-major-tools-full-version-authoring-grid-players-v2-7-4-2.exe having the following md5sum: 70390f7b0e88a4cdbdf8187aac4e3c2e), we have found a number of issues that we would like to point to your

attention so that they can possibly be solved in the next release. I list them hereafter with an indication of the severity of the problem:

1) (blocking) access to the REL translation functions from JavaScript is disabled;
2) (blocking) access to the audio fingerprinting extraction from JavaScript is disabled (error message is <<AxAFPExtract is not defined>>);
3) (blocking) the aac audio format is not supported (in any tool - i.e. the Editor, the Rule Editor, the Player)
4) (annoying) the mime-type currently assigned to the mpeg-1 audio layer III (i.e. mp3) format is audio/mpeg. Instead, the AudioAdaptation.FFAudioTranscoding call of the Rule Editor requires such resources to be tagged as audio/x-mpeg (experimental). This was the old mime assignment to the mp3 format and is now obsolete.
5) (new feature request) the audio fingerprinting extraction function is defined as follows:
STRING AxAFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 nFeatures, INT32 frameSize, INT32 frameShift, INT32 offset);
The last parameter indicates the offset for the fingerprint calculation, however there is no way to limit the length of the fingerprint calculation (the calculation is always performed up to the end of file). Therefore we suggest to modify the call by adding another parameter named "length" which indicates the number of audio samples over which the calculation is performed. The resulting signature would then be: STRING AxAFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 nFeatures, INT32 frameSize, INT32 frameShift, INT32 offset, INT32 length);

Solution to Issue 2
1) The JSRightExpressionTranslator is now enabled.
2) This message means that you have not used the correct syntax: <name of plugin>.<method(....)>
The right call is: AudioFingerprintExtraction.AxAFPExtract(.....)
3) In the SVN is available the new version of AudioAdaptation and RingtoneAdaptation plugins that support the aac format. It was generated by using the latest version of FFMPEG library.
4) This can be fixed by managing both. Now this features are not available for all mimetypes
5) Development of additional feature in progress. New function will be available soon.

Issue 3
I find that if I protect a file I am no longer able to save as an MP21 object.This is true for the script running in the rule editor, and a manual build using the axmedis editor. The file I am trying is large and has a number of resources. It saves OK as .axm. In both the editor and the script, the error message is ' Error: The media data box is larger than how much declaring'. If I try a smaller file in the editor, it does save, but the file cannot be opened. The new executable does execute the rule - only with .axm. With .mp21 I get an error.

JSERROR: 4HOME_v3 (AXRID = axcprule:609f65cb-09b3-4b31-839f-e6cc0377210a): line 84 in FileandObject
Error type: "Not able to open a temporary file"
Flags: UNKNOWN (Error number: 0)

Actually this now also occurs with the rule editor, but NOT every time. So I have been able to produce .mp21 using the rule with the rule editor. It works about half the time (this is where the error print out is from) I have never succeeded to make mp21 with the new rule executor.

This has never happened to the rule editor before. This rule has been stable since before October.

```
//Build (and save) Object
function  BuildAxmedisObject()
{
  var obj = new AxmedisObject();
  var resources = createResourceArray();
```

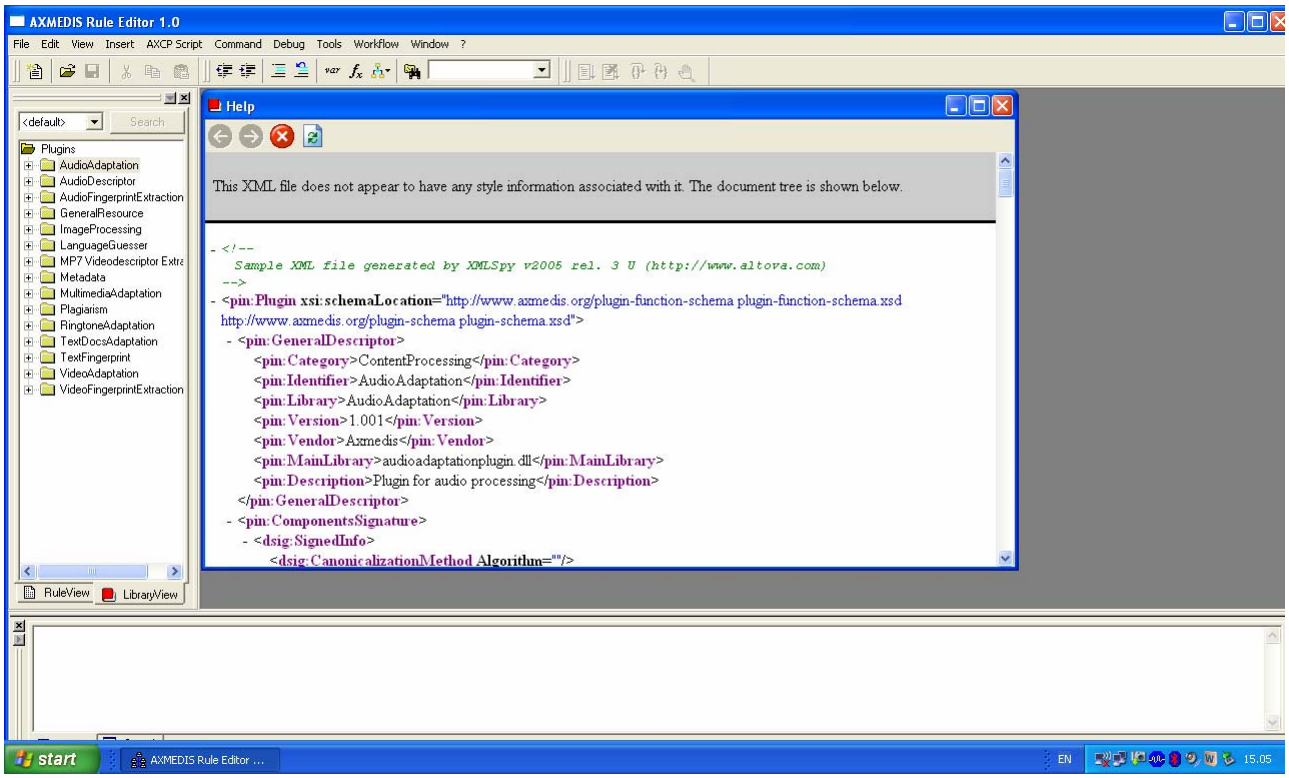
```
if(resources.length==0)
   return false;

for(j=0; j<resources.length; j++)
{
   var resource = new AxResource();
   //print ("   resources["+j+"]="+resources[j]);
   resource.load(resources[j]);
   resource.contentID = resource.localPath;

   // insert the index.smil as first resource for a better visualization
   if(resources[j].indexOf("index.smil")!=(-1))  //was summary.smil
   {
      var reslist = obj.getContent();
      obj.insertContent(resource, reslist[0],true);
   }
   else
   {
       obj.addContent(resource);    ---------------------------------error occurs here.
   }

   resource.flush();
   resource= null;
}
```

Using the rule editor I can see that

1. The files are present in the folder.
2. The files are listed in the resouces array from index [0] to [12]
3. The index.smil file is [4]
4. The rule terminates with the error on the second time around the loop, j=1.
5. The index .smil seems to be properly formed on visual inspection.

Solution to Issue 3
The javascript code seems to be syntactically correct. I suggest to browse the %temp% folder to check the number of files stored in. Try to delete temporary files and try again the rule.

Issue 4
In the new axcp rule editor, if you click on Library/View-->AudioAdaptation-->View Info you get the error shown in the annexed picture.



Solution to Issue 4
Add the following line in the XML file associated with the plugin:

```
<?xml-stylesheet
   type="text/xsl" href="genhtml2.xslt"?>
```

This allows using the stylesheet file for formatting the html page.

Issue 5
I tried both the RightsExpressionTranslator and the OMALicense classes, using the latest version of the AXMEDIS tools (binary release: axmedis-major-tools-full-version-authoring-grid-players-v2-7-4-2.exe), but:
1- the RightsExpressioTranslator class cannot be instantiated (returns an "unknown class" exception);
2- the OMALicense class can be instantiated, but does nothing (has no methods).
So I suppose I have to wait for the next binary release of the AXMEDIS tools...
The translation functionality was omitted in the AXCP. In next version of the executable and AXTools this should be fixed. Anyway, we would need FHGIGD to update the JS classes in order to be able to use the new translation functionality.

Solution to Issue 5
FHGIGD did the neccessary updates: 1.) update js-code, 2.) ruleeditor-support, 3.) ruleeditor-example:
In order to run the JS-example, you need to recompile the axruleeditor. Rule editor has been recompiled according to those changes.

Issue 6
We are having a little problem in producing licenses from rule editor. Just for starting, we would like to produce a distributor + an end user license with the play right only. So we create an object from editor, get (and copy) final AXOID, protect and register to AXCS. Then we put the AXOID into the rule, run and we get error:
License posted as 400: NOT VALIDATED TO PARENT LICENSE, in the PMS Server
In attach our rule and our licenses. Could please someone have a look and tell us what is wrong? We followed the “usual” samples for licensing generation but it seems we make an error somewhere…

Solution to Issue 6
We have been testing the script with ruleeditor and also your licenses directly to PMS. The licenses are totally correct and work fine. What is failing is the following: in the script you are creating and posting both licenses in a different manner.

1) licenses[0].sendLicenseToPMS(); (in createMLicense in ProduceMotherLicensesForContent)
2) postLicense(lic,PMSClient_EndPoint,PMSClient_Certificate,PMSClient_CertificatePsw,PMSClient_CA); (in ProduceEndUserLicensesForContent)

What happens is that 1) is not storing the mother license in PMS, and then 2) fails to post the end user license because the mother license is not present in PMS.
For us, the correct way to post licenses to PMS is 2). After consulting jslicense code, I see that 1) is also possible. It was probably added by FHGIGD to JS classes. In fact, it is internally using the same code (instantiating pms client and calling postLicense). For using 1) you need to pass the appropriate arguments to sendLicenseToPMS method (char* endpoint, *ClientCert, *PassClientCert, *CACert;)
So you can choose to use 1) or 2). Anyway, you have to change the code in the script.

5.4.2 AXEPTool

Issue 1
I have installed the new AXEPTool (v. 3-2-33), now when I start it I get the following error message from the JVM: "Could not find the main class. Program will exit."
However the AXEPTool starts and runs without problems - it seems that the message is irrelevant.

Solution to Issue 1
Update the JVM

Issue 2
I have found out that is possible to have different versions of the JRE installed on the same machine. Now on my ax4home server I have both jre1.5.0_12 and jre1.6.0_03 under the C:/Program Files/java directory.
I have no problem to configure tomcat 5 so that it uses the 5.0 branch (it only needs the JAVA_HOME environment variable to be set), but I have no idea how to say to the AXEPTool to use the 6.0 version of the JRE. Currently the AXEPTool continue to use the 5.0 branch (and fails). Do you know how the AXEPTool decides which jre to use?
AXEPTool chooses the java current version to use. So look in the registry HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment and you'll find a key entry called CurrentVersion, that is the java used by the tool. You can edit it to allow the AXEPTool running with java 6.
The entry in the registry is correct (current version = 1.6) but the AXEPTools ignores it due to the environment variable JAVA_HOME being set to C:\program files\java\jre1.5.0_12;

Solution to Issue 2
The problem was solved by TI by removing the JAVA_HOME definition from the system environment on WinXP, and defining it locally in a script that is called whenever Tomcat starts. The script is named setenv.bat and contains just the following definition:
set JAVA_HOME=C:\Program Files\Java\jdk1.5.0_12
Now the AXEPTool is running on java 6, while Tomcat 5 still runs on java 5 on the same machine.

5.4.3 DRM Editor and Viewer

Issue 1
With DRM editor (intergated in axeditor)
1. I have created a new license without selecting the issuer panel

2. the reply when sending to PMS server is "OK. Logs in Secure Cache."
3. looking into the PMS Server log the upload failed for the issuer different from axuid in the certificate and in the XML received the issuer is missing.
4. then I select the issuer element in the license tree structure
5. send to the PMS Server ("OK. Logs in Secure Cache")
6. looking to the Server logs it failed for not validated with parent license

So it seems that the issuer is inserted only if I select the issuer element. The reply when uploading the license is always OK. How can the user see the Logs in secure cache?
I get the NOT VALIDATED TO PARENT LICENSE even if the licenses are correct, with the old version of PMS Server licenses like the one I'm tring to add were good. However licenses that are automatically created using the PMSClient functionalities are good and working.

Solution to Issue 1
The problem about the "issuer" has been fixed and the files uploaded. This solves most of the problems. Messages to the user are still pending of a review. It still has to be decided which information of SecureCache should be available for the user.

Issue 2
The PAR editor behaves in a strange way:
- create distribution license
- add a grant
- remove the grant
result: it removes the issue grant

The PAR editor crashes doing this:
- create distribution license
- add a grant
- add another grant
- remove one of the two grants.

Solution to Issue 2
DRM Editor and Viewer has been updated to solve the bug.

5.4.4 AXMEDIS ActiveX

Issue 1
The last version of axactivex integrating the SMIL+HTML has a bug (an assertion failed) in the CWinApp::SetCurrentHandles() method. I think this is due to the MFC classes used for the HTML display in the SMIL player. However pressing cancel to continue then the player seems to work. The dialog pops up only when opening a SMIL with html.

Solution to Issue 1
A new version of the axactivex in release has been produced. It does not pop up the dialog.

Issue 2
We have tried to generate distributor and end-user license from DRM Editor inside AXMEDIS Editor. Rights for play and modify, since we would like to grant the possibility of modifying an object even if it is protected. This is possible, isn't it?
We can create and save on server the mother license, but when we try to store the end-user licence on server we get the error:Ok. 400: license_rejected, Not validated to parent license,

In attach:
- "clear object" (not protected)

- Protected object
- Mother license
- End user license
- Screenshot of the error

Please don't' hesitate to contact us if you need further information to understand the reason of the error.

Solution to Issue 2
The problem is the next:
In your distribution license the principal of the "issue" right is
URN:AXMEDIS:00002:BUS:92CDC983-1D9B-3E9F-B96B-F4FA8655C6FF
And in your end user license the issuer is:
URN:AXMEDIS:00002:BUS:58EDD4C1-F931-37D9-90A5-BEDCCAAF0846

The issuer of the end user license must be the entity that has the "issue" rights. And that's the problem. Another thing, I have noticed that in your distributor licenses inside each "enduser" grant you have also inserted the principal. This will suppose that you can only create end user licenses for this user. If in the distributor license inside the enduser grants (play and/or modify) you don't put principal, you will be able to generate end user licenses for any user you need. But, always the principal of the issue grant must be the same of the issuer in the enduser license.

5.4.5 PDA player

Issue 1
When trying to open a protect object, the AXPDAPlayer says: "Load Device verification failed when opening a protected object". But then it opens successfully the object. In attach the object.

Solution to Issue 1
The problem is due to an uninitialized buffer in the function used to get the system ID which produces random behaviour. A new signed exe and installer has been produced.

5.4.6 AXMEDIS Editor, Player and Skin Player

Issue 1
What we would like to achieve is:
- create axmedis objects,
- protect them
- create licenses (mother and end user) in order to grant the possibility to modify these contents later

Is this possible? Which rights do we have to use? And which "modifications" (via AXMEDIS Editor and/or AXCP) are possible? Since we did a test:
- create content
- protected
- given play and modify rights

then:
- ok play from the player, but
- from the editor, nothing is editable, we can see only the axinfo, even the (unique) resource of the object is not playable.

Solution to Issue 1
After registering a protected object to AXCS, the protection information is removed from the object and registered in AXCS. The editor asks you whether you want a distributor and end user license to be created

for the right play. In this way, you can open the object with the player or editor (you will be authorised and you will get the protection info from AXCS). In the editor the protected object is opened double-clicking on "Protected object [URN....]". Double-clicking asks for the "play" authorisation, which is a right that you have if you accepted to create the play license before.

Regarding the object edition, I don't know which is the current status. I have opened a protected object and I have been able to add more resources, so I think it is not checking the rights (asking for authorisation) yet. DSI can provide more details about this aspect.
What you were missing was double-clicking on the root to get the hierarchy visualization. After that you can also add resources and modify the Dublin Core.

Issue 2
The problem I find is when updating with the final AXOID that AXCS has sent. I made it work well in my computer, but with a non-elegant solution:
- In method "AxWizardPanel::onFinalAxoid" (line 160) I call:

```
AxEditorFrame* fr=dynamic_cast<AxEditorFrame*>(frame);
//new from UPC
AxEditorPanel *aep= (fr ? fr->getEditorPanel() : NULL);
//new from UPC
if (aep) aep->panelDRM->RefreshAXOID();
//new from UPC
```

- What makes me include also "axdrmview.h" and "axeditorpanel.h" at the beginning.
- In addition, "axeditorpanel.h" has to change the declaration of panelDRM and make it public.

This works, but I find it unacceptable, as it violates all the rules of elegance in the coding: "panelDRM" should stay private, and no other class rather than AxEditorPanel should deal with the DRM E&V classes.
Furthermore, the method in "AxDRMView.h" does not fit well to the philosophy of the class. Therefore, I have not uploaded the changes in DRM E&V classes to do this, nor proposed you this solution.

Have you dealt with any similar problem? (i.e. a module which needs to be updated with information of axeditor or other modules) Could be a solution using messages?

Solution to Issue 2
If you expose the functionality to update the AXOID I will call it in the proper place.
I will restructure the code using the functionality to request the AXOID of the axhierarchyeditor (where it should be) and I will call the refreshAXOID on the DRM editor in the AxEditorPanel.
Unfortunately the AXOM events seems to not catch the AXOID modification as an event, I will investigate why, but it is a quite complex piece of code written by Leonardo when he was working with us.
You can commit the changes to AxDRMView and I will do the rest. When I will find why the AXOM events do not work for the AXOID update we can use them to call the refreshAXOID() method directly.
Status: implemented

Issue 3
I tried putting some of the scripts directly into the editor to see if they can be executed without the rule executor. Unfortunately it seems not to have some of the functionality we need, eg, accessing raw files from the disk to create the resource. Is this still work in progress or a more basic limitation?

Solution to Issue 3
Enabling all the capabilities of AXCP rules on a player used by the final user may put the user's computer to the risk of malicius use (virus). A version of the player/editor enabling all the capabilities only if the user explicitly enables it will be created.

5.4.7 Protection Processor, PMS Client, PMS Server, AXCS and AXMEDIS CA

Issue 1
We are trying to use the PMSCLient for mobile, we succeded in using certify, but we expected that it provides the AXTID as in the PC. The AXTID is needed also as parameter for Verify and Authorize, where can we get AXTID? Moreover the interface of the authorise seems to not allow to access the protection info.

Solution to Issue 1
PMSClient and PMS Server for mobile have been changed so that:
- authorise returns Protection Info in result
- certify returns AXTID and Enabling Code in result

Issue 2
I'm experiencing problems with axmedis activex and securecache. The activex initializes and deinitailizes more than once, this means that the PMS client is constructed, destructed and then reconstructed again. After the second construction opening securecache hangs the application and after 1 minute it fails.
I try to explain what happens:
0. a window (IExplorer page) with activex is opened
1. the activex is initialized and the ProtectionProcessor is initialed
and thus PMSClient is initialied and thus the SecureCache is initialized
creating the instance in the singleton
2. the activex control is created and it works
3. the window containing the activex is closed (but not the application)
4. the activex is terminated and the ProtectionProcessor::terminate is
called which deletes the PMSCLient (which does not terminate the
securecache)
5. the window with the activex is opened again
6. the activex is loaded again and initialization is done (in this case
also all the global static variables are initialized and thus the
pointer to the securecache in the singleton is set to NULL) this leads
to create another instance of the securecache and it waits a lot of time...

I tried to add to the secure cache a method to destroy the instance and to close the db (I called it from the PMSClient destructor) but it seems to not solve the problem.

I noticed a similar problem with the player, the player opens one axobject at a time, if you double click in explorer on a .axm file it opens an axplayer.exe and if you open another .axm file from explorer to start the second instance stops until I close the other player.

Solution to Issue 2
The problem is due to the fact that the SecureCache, aimed at simplicity, stores the database in a single file, non-concurrent by nature. Thus, two players may not operate at the same time. Solving this, would imply opening and closing the database in every database access operation. I think this would solve the problem but the patch is not trivial.
Status: pending to be implemented

Issue 3
We have implemented the enforcement for B2B operations on protected objects like embed, adapt etc.
An operation requested by the enduser may result in a series of grants to be checked. For example to copy a protected object inside another protected object should request:
- the grant to embed the object inside another one
- the grant to adapt the object
- the grant to enhance the object where the other object is embedded

The current PMSClient interface allows to check one right at a time and it generates an actionlog for each request. But if for example I have only the right to embed and adapt an object but not to enhance the another

one, the operation fails, but in the action log may be present requests that have not been really exploited by the enduser. For this we should think to a transactional request.

Solution to Issue 3
To support REL Dibos for DIP we have implemented a new functionality that, perhaps, will be useful to make these checks. The function will be a "fake" authorise, that will make the authorisation algorithm, but it will not send and store any ActionLog and it will not return any protection information. This function will have a very similar interface than authorise but, it will only check if there are available rights (licenses) that allows you to grant a "real" authorisation.
You can use this function to make 3 different calls, and if all the calls return a positive (fake) authorisation, you can then call the real authorisation process (3 times in this case).
In this way you can be sure that when calling authorise the result will be positive, and then all Action Logs will be correct.

Issue 4
We have been working to solve the problem regarding a user creating a license in name of another user from any tool. The best and easiest way to perform this check is to compare the issuer in the license to the AXMEDIS User ID present in the certificate used to establish the SSL channel between the client and PMS Server. GSOAP enables to access the certificate information, so this can be easily solved. However, we need to be sure that the certificate used to establish the SSL session is the user certificate instead of the tool certificate. Otherwise it would not be possible to access the User ID.

Solution to Issue 4
The protection processor has been updated to connect to PMS only using the User certificate. However, there is also another case to be considered: when a valid business user is creating an issue license for an object that he has not created. In this case any business user can create an issue license for any axmedis object. Maybe you don't have enough information to perform the check at license creation (unless you ask it to AXCS), however during the authorization phase you have this information.
The only solution for this is to check the content ownership against AXCS, as you suggest (or even use the ontology...). However, if we solve the issuer problem, ensuring that the issuer in any license is correct, then if someone issues a license for some content for which they are not allowed, Action Logs will be registered and they could be prosecuted.
Status: not implemented

Issue 5
During these months that I have used the PMS/AXCV I have found frustrating the fact that was difficult to understand the reasons why verification was failing. Looking to the PMS server log file helped me a lot of time (however some important information was missing like the tool information in reverification, needed to understand what was changed in the hardware/software that produced a verification failure)
The logging in AXCV is done to the standard output that is mixed WITH A LOT OF OTHER LOGGING MESSAGES and finding the message related with the error I was getting was difficult (the messages have not a prefix that I can use to filter the log).
Considering that in some months the DRM tools will be used by "real" end user and if a "customer" asks me "why my user has been blocked?", or "why I get verification error?" to reply to him "You have changed your CPU" I need to browse a log file looking for his axuid trying to understand what happened... (and in some cases the information is missing).
I suggest:
1. to make a separate logging for axcv reporting all the events with all the information received
2. to store important events like user/tool blocking in a database to allow to find events related with an user
3. in case an important event happens (user/tool blocked) to send an email to an administrator that can find and solve the problem before someone complains.

Solution to Issue 5
1. A more detailed logging has been provided for AXCV.

2. This is stored in Action Logs in AXCS database. You can consult them and serach by any of the fields.
3. An emailing mechanism has been implemented, It can be configured in AXCV configuration file (axcv.properties)

Issue 6
I'm tring to provide to user some more messages when the authorization fails, in the documentation of authorise the return value for failing authorization is not completely specified between -3001 and -3128 (and similarly between -1001 and -1128 and -2001 and -2128). Please can you tell me the meaning of these errors?
I think that the following are errors meaningful for the user:
- number of executions passed
- not in the proper territory
- the time limit is passed
- right is missing
- no license

Solution to Issue 6
The authorisation support returns a bitwise information about the results of the authorisation:
#define UNKNOWN_ERROR 1
#define LICENSE_REJECT 2
#define ERROR_OPENING 4
#define DATE_INVALID 8
#define CANNOT_CONNECT 16
#define OUT_OF_DATE 32
#define NO_MORE_TIMES 64
#define TERRITORY_INVALID 128

00000000 --> HGFE DCBA

If A = 1 --> UNKNOWN_ERROR,
If B = 1 --> LICENSE_REJECT (there is no license that proceed with the authorisation)
If C = 1 --> ERROR_OPENING (error opening the database)
If D = 1 --> DATE_INVALID (if the licenses has a issue time forward than now, for avoiding spitefull uses of system clock).
If E = 1 --> CANNOT_CONNECT
If F = 1 --> OUT_OF_DATE (if exist one license,with an interval condition, and you are out of the interval)
If G = 1 -->NO_MORE_TIMES (if exist one license,with an count limit condition, and you have consumed all the executions)
If H = 1 -->TERRITORY_INVALID (if exist one license,with an territory condition, and you out of this territory)

So, if you get a 160 --> 1010 0000 --> you are not authorised, but you have found a license for the user, right and resource, that doesn't allow you the reproduction because you are out of the interval and out of the territory.

Issue 7
I have installed the PMSGateway on violino (where it is present the PMS Server, and AXCV). I have added a connector on port 8445 to be used to get. When I try to certify the mobile I get -30 as result (internal AXCV error).

In the PMS server log is present:
certifyForMobile(URN:AXMEDIS:00002:USR:8DFCF74C-0BCF-3B9E-8976-AF2F13DEFB9B,,URN:AXMEDIS:00002:RTO:0000001A-0000-0000-0000-00000000002F,<?xml version="1.0" encoding="UTF-8"?><ToolFingerprint><DeviceFingerprint><MobileDevice><IMEI>NOT

PRESENT</IMEI><PhoneNumber>+399999999</PhoneNumber><JavaPlatform>SunMicrosystems_wtk</JavaPlatform></MobileDevice></DeviceFingerprint><SoftwareFingerprint><FileFingerprint><Digest>NOT PRESENT</Digest><CreationDate>NOT PRESENT</CreationDate><LastModificationDate>NOT PRESENT</LastModificationDate><Size>2458</Size></FileFingerprint></SoftwareFingerprint></ToolFingerprint>)=-30

and in the tomcat logs is present:
AXCV Internal error:javax.net.ssl.SSLHandshakeException: Received fatal alert: bad_certificate
OR
AXCV Internal error:java.net.SocketException: Software caused connection abort: recv failed

what can be the problem?

I'm using the old PMS server on port 8502
I have found that having the tomcat on https on port 8445 produces this behaviour.

Solution to Issue 7
Having the PMSGateway in the same Tomcat where AXCV is deployed causes a communication error between AXCV an AXMEDIS CA (EJBCA on JBoss in the same machine). The solution consists on deploying PMSGateway in a different machine with different Tomcat. It is still unknown why such conflict occurs.

Issue 8
I have noted that if I use the authorise with a wrong hwfingerprint it gives the protection info and blocks the user, it should not provide the protection info.

Solution to Issue 8
PMS has been updated to solve the bug.

Issue 9
I have a small error in the error reported from PMS. I tested the number of uses of a right, I added a licence with 5 as number condition and at the sixth use the error report -3002 which means that no play license is present, while I expected a -3064 meaning that no more excecutions are left.
I added a new license with 3 more uses and it behaves correctly allowing me to play three more times and at 4th it tells that no play license is present.
It is not a big problem but now I provide to the user a description of the error and it would be nice to tell to the user that he finished the executions.

Solution to Issue 9
Due to the internal operation of the authorisation method, it is not easy to distinguish between those two cases: "no play license is present" and "no more excecutions are left".
The authorisation process uses a list of licenses to see if any of them authorise the request, i.e. all licenses for the user, resource and right in the request. This means that some of them may fail because of time restictions, some others because of other conditions and some others because the max number of times have been consumed. It is not easy to say why the authorisation has failed, when each license may fail because of different reasons.

However, at least we could distinguish two different cases:
- There is no license for the request
- There are some licenses and authorisation fails because the total number of times (taking into account all licenses that fulfill all the conditions) has been exceeded. In this case some other licenses may fail due to other conditions such as time, location, etc. We suggest in this case just to aware the user that "no more executions are left".
Status: not implemented

5.4.8 AXMEDIS Tools

Issue 1
Our machines are all behind proxies, this means that to access https (like in the case of PMS) we have to go through proxy. Thus it is impossible to certify a machine, because it will just not connect. Anyone has experienced this situation already? Possibly solving it? Otherwise I think the possibility for setting a proxy for internet connection(s) triggered by the tools should probably be provided.
Firewall support: currently the PMS server communicates with clients by using the SSL protocol on port 8502, which is not the default SSL port (443). Most firewalls block the SSL exchange when it is executed on a port different from 443. Therefore it would be more appropriate to use the default SSL port.

Solution to Issue 1
PMSClient and other tools have been modified to allow their calls to external servers through a proxy (e.g. PMS Server).

Issue 2
I am still baffled by the behaviour of the latest tools when using the PMS. I am afraid I simply cannot use them in conjunction with the PMS. I have tried to organise my tests to give you definitive behaviour. Here are the results.

Summary:
I think the rule editor is unstable and cannot post a licence (new portocol?)
The new editor and player also fail to find the licence at the new PMS address.
The old tools do locate the licence at the new PMS address
Nothing connects to the old PMS (:443) - which is OK I guess.

What works:
Oct 11 2007 Rule Editor with PMS set to :8502/pms/ in script and configuration

Rule executes as follows
Creating Dublin Core Metadata
URN:AXMEDIS:00002:OBJ:883FEC8E-F6DF-3E84-8D67-2EB4E6DDCC3B
tempDir: Z:\\4HOME_temp_output_path
Registering object to AXCS
Start Posting Mother License on PMS
License posted as 200:LID3008 in the PMS Server
License posted as 200:LID3009 in the PMS Server
Return: true
Execution terminated

Then;

Player
nov 20th - OK when pms changed to :8502/PMS/
.net OK when configuration updated

Editor
built nov 20th - can open and play once PMS has been cahnged to
:8502/pms/ - OK

What doesn't work:

Player
Dec 27th version of player cannot find licence or to play 'Unknown

Exception' Configuration says pms is correct

Editor
Jan 03 2008- 'unknown exception' NB it has just certified tool, so
connection to PMS is live. changed from:8502 to :8502/pms/ - same error.

RuleEditor
Dec 10 2007 Rule Editor with PMS set to :8502/pms/ in script and
configuration Crashes. - when posting licence
Dec 19 2007 Rule Editor with PMS set to :8502/pms/ in script and
configuration Crashes - when posting licence

Solution to Issue 2
Regarding AXCP tools, the JS classes and scripts have not been updated to deal with the new version of PMS. As we informed in a previous mail, the new version of PMS means that old tools would not continue working with this new version so all tools and scripts would need to be adapted or continue using old PMS. Changes were made according to the requirements in the action plan.
At the date of closure of this deliverable, JS classes have been already updated

5.4.9 AXMEDIS User Registration portal

Issue 1
I noted the following problems while performing the AXMEDIS user registration procedure:

- (annoying) the user registration procedure doesn't check for duplicated nicknames inserted by end users in the registration web page. Instead, the procedure accepts the data, then sends an email to the end user containing an hyperlink to the registration portal; however when the user clicks on the link, if the nickname was a duplicated one, the result is a misleading error message: "server error, please try again later". Trying multiple times always results in the same error message being displayed.
- (annoying) the user registration procedure does not allow the end user to set its own password. A random password is generated, which is very difficult to remember.
- (annoying) the user registration procedure sends a total of three email messages to the end user to complete, when one would be sufficient.
- (annoying) the user registration procedure requires all fields to be filled (even the fax number! not quite common for end users).

Since the user registration procedure is one of the first operations with which a new AXMEDIS user is confronted, and given that improving these parts shouldn't be a big effort (well, I hope so... but you know better than me), I think it would be beneficial to the project to solve these issues.

Example of registration steps:
#1 from web page
Dear TILABdistributor,
your request has been accepted, you will receive an email at the address email@url.it with a link. Follow the link to confirm your registration.
#2 email
Click the following link to confirm your registration:
http://axcs.axmedis.org:8080/RegistrationPortal/confirm.jsp?id=email@url.it&uuid=5D9CBB11-954F-3A43-A02F-025B3B33056D
#3 from web page
Your AXMEDIS UserId is: URN:AXMEDIS:00002:BUS:5D9CBB11-954F-3A43-A02F-025B3B33056D and your password is: xxxxxxxxx.
You will receive at the email address email@url.it with the confirmation of the subscription.

#4 email
Dear UserName,
welcome to the AXMEDIS system.
Your registration is completed successfully and you can find in attach your AXMEDIS personal certificate.
Don't loose it!!!
Regards,
AXMEDIS Consortium
#5 email
Dear TILABdistributor,
You have been registered on AXMEDIS with AXUID URN:AXMEDIS:00002:BUS:5D9CBB11-954F-3A43-A02F-025B3B33056D and password xxxxxxxxx
AXMEDIS Team

Solution to Issue 1
Solution: the user registration procedure should check for duplicated nicknames in advance, and inform the end user immediately (possibly proposing a new nickname).
Status not yet provided neither by EXITECH nor by DSI

6 References

AXMEDIS-DE5-2-1-AXMEDIS-Framework-Validation-v2-1.pdf
http://www.axmedis.org/documenti/view_documenti.php?doc_id=1392

AXMEDIS-DE5-2-2-AXMEDIS-Framework-Validation-and-Integration-Report-v1-0.pdf
http://www.axmedis.org/documenti/view_documenti.php?doc_id=2447

AXMEDIS-DE5-2-2-2-AXMEDIS-Framework-Validation-and-Integration-Report-v1-0.pdf
http://www.axmedis.org/documenti/view_documenti.php?doc_id=3482