



Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE3.1.2A

Framework and Tools Specifications (General Aspects and Model)

Version: 2.1

Date: 17/03/2005

Responsible: DSI

Project Number: IST-2-511299

Project Title: AXMEDIS

Deliverable Type: Private

Visible to User Groups: NO

Visible to Affiliated: NO

Visible to the Public: NO.

Deliverable Number: DE3.1.2, Part A

Contractual Date of Delivery: January 2005

Actual Date of Delivery: 17 march 2005

Title of Deliverable: Document

Work-Package contributing to the Deliverable: WP3.1

Task contributing to the Deliverable: WP3, WP2

Nature of the Deliverable: report

Author(s): DSI, EPFL, FUPF, ILABS, CRS4, FHGIGD, DIPITA, IRC, XIM, HP, EUTELSAT, SEJER, TISCALI, COMVERSE

Abstract: This document contains part A of the specification for the first 18 months of the AXMEDIS project. In this part the general aspects are reported like the specification guidelines, the implementation guidelines, the general architecture. This part also reports the specification of the model used for AXMEDIS objects, both in memory and as MPEG21 digital item and for the specific metadata associated with the content and with the whole object. It also reports the specification of the tools strictly related to AXMEDIS objects management like the AXMEDIS Editor and the AXMEDIS Object Manager as well as the tools for content protection management.

Keyword List: model, metadata, MPEG21, protection, DRM

AXMEDIS Copyright Notice

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

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

1. DEFINITIONS

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

2. LICENCE

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

3. TERM AND TERMINATION

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

4. USE

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

5. COPYRIGHT NOTICES

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

6. WARRANTY

DE3.1.2A – Framework and Tools Specification (General and Model)

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

7. INFRINGEMENT

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

8. GOVERNING LAW AND JURISDICTION

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

Please note that:

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

Table of Content

1	EXECUTIVE SUMMARY AND REPORT SCOPE (DSI, ALL)	6
2	SPECIFICATION GUIDELINES (DSI, ALL)	7
3	IMPLEMENTATION GUIDELINES	9
3.1	DECISIONS ON DESIGN, IMPLEMENTATION AND SPECIFICATION TOOLS (ALL)	9
3.2	IDENTIFICATION OF A FRAMEWORK FOR CROSS PLATFORM C++ APPLICATIONS DEVELOPMENT TO BE USED AS BASE OF AXMEDIS FRAMEWORK	10
3.2.1	Requirements for the C++ Framework	10
3.2.2	Candidates for the C++ Framework	10
3.2.3	Evaluation of Candidates	11
3.2.3.1	Conclusions	15
3.3	SUPPORTED PLATFORMS (ALL PARTNERS)	15
3.4	DEVELOPMENT FRAMEWORKS	16
3.5	USER INTERFACES (ALL PARTNERS)	17
3.6	COMMUNICATION (ALL PARTNERS)	17
3.7	INFORMATION EXCHANGE (EXITECH, ALL PARTNERS)	17
3.8	CVS REPOSITORY (EXITECH)	17
3.9	CODING GUIDELINES	19
3.10	DOC GENERATION	22
3.11	ADOPTION OF LIBRARIES	22
3.11.1	Licensing terms (All Partners)	23
4	AXMEDIS GENERAL ARCHITECTURE	24
4.1.1	AXMEDIS General Architecture	27
5	AXMEDIS FRAMEWORK OVERVIEW, GENERAL ARCHITECTURE (DSI, ALL)	31
5.1.1.1	WP5.1 -- Development of the general infrastructure	33
5.1.1.2	WP5.2 -- Component Validation and Acceptance	33
5.1.1.3	WP5.3 -- AXMEDIS framework integration and maintenance	34
6	AXMEDIS EDITOR OVERVIEW (WP4.1.3, WP5.4.4: DSI, EPFL, FUPF, ALL)	35
6.1	SOFTWARE ARCHITECTURE	38
6.2	GRAPHIC USER INTERFACE	39
6.3	CLASS DIAGRAM	41
7	AXMEDIS OBJECT MANAGER (DSI, EPFL)	42
7.1	AXMEDIS COMMAND MANAGER (DSI)	44
7.2	PROTECTION PROCESSOR	47
7.2.1	AXMEDIS tool registration and certification	48
7.2.1.1	Software and hardware fingerprint	48
7.2.1.2	Tool certificate	48
7.2.1.3	User certificate	49
7.2.1.4	User Identifier/Identification	50
7.2.1.5	Date and time	50
7.2.1.6	Action history	50
7.2.1.7	Enabling code	51
7.2.1.8	Trustiness of a tool	51
7.2.1.9	Certified software	52
7.2.1.10	Execution controls	53
7.2.2	Robustness against malicious user actions	53
7.2.3	Protection Processor Class hierarchy	54
7.2.4	Protection Info and Procedure Interpreter (DSI)	57
7.2.4.1	Protection Info Format (DSI)	58
7.2.4.2	Protection Interpreter (DSI)	58
7.2.4.3	Protection commands (DSI)	59
7.2.5	Tool certification/registration (DSI, FUPF)	60
7.2.6	Tool verification/authentication (DSI)	62

7.2.7	Tool Fingerprint Estimation on PC (DSI)	66
7.2.8	Tool ID Estimation on PDA (EPFL)	69
7.2.9	Scramble/Descramble Support (EPFL)	71
7.2.9.1	cryptlib	72
7.2.9.2	How to install Cryptlib on Windows XP	73
7.2.9.3	How to use cryptlib in C	73
7.2.10	Compress/uncompress Support (DSI, lib prob.)	75
7.2.11	Encryption/Decryption Support (FUPF)	76
7.2.11.1	Architecture for encryption / decryption support	76
7.2.11.2	OpenSSL	79
7.2.11.3	Windows Version of OpenSSL	79
7.2.11.4	Cryptographic functions provided by OpenSSL	79
8	AXMEDIS DATA MODEL SUPPORT (WP4.1.2: DSI, WP5.4.3: DSI, EPFL)	80
8.1	AXMEDIS DATA MODEL SCHEMA (DSI)	81
8.1.1	AXMEDIS Objects as MPEG21 Objects	81
8.1.1.1	MPEG21 Digital Items	81
8.1.1.2	AXMEDIS Objects	83
8.1.1.3	Basic AXMEDIS Object:	85
8.1.1.4	Protected Basic AXMEDIS Object:	87
8.1.1.5	Composite AXMEDIS Object:	87
8.1.1.6	Protected Composite AXMEDIS Object:	89
8.1.1.7	Governed AXMEDIS Object:	90
8.1.1.8	Query/Promotional AXMEDIS Object:	90
8.1.2	AXMEDIS Metadata Model (DSI, EPFL,)	91
8.1.2.1	AXInfo Model	91
8.1.2.2	Dublin Core Metadata	110
8.1.3	Examples of AXMEDIS Objects	115
8.2	AXMEDIS MODEL (DSI)	124
8.2.1	MPEG21 Digital Item Model	125
8.2.2	AXMEDIS Object Model	129
8.2.2.1	AxMetadata	131
8.2.2.2	AxInfo	132
8.2.2.3	AxDublinCore	135
8.2.2.4	AxOID	136
8.2.2.5	AxContent	137
8.2.2.6	AxObject	137
8.2.2.7	AxResource	137
8.3	XML LOADER AND SAVER (DSI)	138
8.3.1.1	Xerces-C++	144
8.4	AXMEDIS OBJECT PREPROCESSOR AND POSTPROCESSOR (EPFL)	144
8.4.1	XML to/from BIN (EPFL)	145
8.4.1.1	BIM	145
8.4.1.2	Xmill	146
8.4.1.3	BinXML	147
8.4.2	References In/Out Resolver/Integrator (EPFL)	148

1 Executive Summary and Report Scope (DSI, all)

This document reports the specification for the first 18 months of the whole project activities. It takes information from the WP2 of user requirements and from the preliminary work performed by project partners and summarised in this proposal document. The specification take into account the general structure of the AXMEDIS framework and of the demonstrators planned in WP9.

The specification is focused on mapping the research and development work on the real needs provided by the requirements, use cases and test cases:

WP3.1.2 -- Specification of AXMEDIS framework (M3-6) – managed by FUPF – mainly used for preparing WP5: specification of AXMEDIS framework and major common tools (authoring, distribution support, protection support, composition and formatting program definition and execution, etc.) (mainly used in WP5); considering: AXMEDIS Multimedia modelling, AXMEDIS general architecture and intermodule communication; P2P, Data sharing, streaming, event synchronisation, indexing, DRM, DRM interoperability, protection, monitoring, fingerprint, certification, etc.; Distribution channels; content player tools and paradigms; Acceptance and validation testing processes, etc., major common tools (authoring and production tools, aggregation tools, formatting tools, distribution support, protection support, etc.) (this information will be mainly used in WP5), AXMEDIS Certifier and Supervisor, AXEPTtool specification, etc.

WP3.1.5 – Specification of tools for automatic content production and formatting – managed by DSI – Specifically defined for WP5 tools on content composition, transcoding and formatting, specification of the main tools for implementing and supporting content composition, formatting and aggregation etc., considering: technical aspects, DRM, content description, metadata, indexing, synchronisation, formatting style, graphic details, devices, user profile for content on demand, final format, distribution tools at the state of the art, production tools at the state of the art, physical models, etc.

The full document has been decomposed in several parts:

- A. This document with general aspects up to the description of the content model (DSI)**
- B. Viewers and players, including plug ins, etc. (DSI)**
- C. Content Production tools and algorithms (DSI)**
- D. Fingerprint and descriptors algorithms and tools (FHGIGD)**
- E. Database area, query support and Content Crawling from CMS (EXITECH)**
- F. AXEPTool area, for B2B distribution and Programme and Publication for B2C distribution (CRS4)**
- G. Workflow aspects and tools (IRC and HP)**
- H. Protection tools and support, Certification and Supervision and Accounting tools (FUPF)**
- I. Distribution tools and AXMEDIS Portal (EUTELSAT)**
- J. Definitions, tables, terminology, acronyms, lists, references, links and Appendixes (DSI)**

This document (part A) reports the general aspects related to the specification task like the specification guidelines and the implementation guidelines. It also reports the specification for AXMEDIS objects modelling and the specification of the tools strictly connected with the model like the AXMEDIS Object Manager which is the interface used by all applications to access and manipulate an AXMEDIS Objects. In this sense it also deals with protection aspects since the AXMEDIS Object Manager has to protect access to the content on the basis of digital rights management policy defined by the content owner/distributor.

2 Specification Guidelines (DSI_{all})

The whole AXMEDIS system has been decomposed in subsystems and tools. The decomposition has been performed on the basis of structural aspects, the diagrams are reported in the UML file in vision. Please see the last version on the Specification folder on the web portal.

The specification of each tools, component and/or module has to be performed by providing the following information and adopting the UML methodology and tools/diagrams this will allow to talk a unique language for all now in the specification phase and in terms of documentation of companies accessing to take-up actions:

- General description of the functionalities and relationships with other tools and components.
- References to the other tools and components that have to interact with the entity.
- Structural decomposition of larger modules or subsystems that still needs to be decomposed to identify the entities that are modeled in terms of classes. These are recognizable being single programs, DLL, plug-in, packages, etc.
- Class diagram with details regarding specialization, interfaces, decomposition and references
 - Description of classes with their major attributes and methods, with their type and signature
 - If some class/object has some evolving state please provide a state diagram with the description of the states and transitions.
- Object diagrams (component diagrams of UML) to show what happen among objects when these are instantiated from classes, to highlight the production of lists, and the general structure of objects in the memory.
- Sequence diagram and/or collaboration diagram (among processes) of UML for selected parts to the explanation of the entity behavior and their relationships with other entities or processes
- Description of protocols, if any, at level of communication packets and all the higher levels
- Description of relevant algorithms for the functional part of your methods/services by using: textual description highlighting the motivation and the needs in AXMEDIS, description in terms of flow chart or activity diagrams of UML or pseudocode or directly in programming language, and if rational a mathematical formulation of the algorithm or of its math parts. For each non specified algorithm since it is not know yet please provide
 - metrics for its evaluation
 - reference value of these metrics
 - an example of the results
 - test cases have to be put in a different deliverable
 - etc.:
- Description of the API provided, if any, in terms of functions/procedures, functionalities, parameters, types of parameters, behaviour, and internal behaviour. In addition, a sample procedure and detailed sequence diagram of what can be done to exploit the module in other processes. The API has to be described by using UML VISIO, IDL (interface description language).
- Description of the interoperability specification aspects related to the adoption of the software module in different operating systems and to be integrated in different contexts,
 - conditional compilations,
 - different behaviors in different context,
 - profiling,
 - configuration aspects,
 - etc.
- Formal description of any textual format file, all content formats and confirmation formats have to be XML and have to be provided in terms of Schema, where each field has to be fully specified in terms of type and semantics of each possible value, giving the dynamics (e.g., -255 + 256), type (e.g., string, float, integer, unsigned integer), etc. Some examples have to be provided.
- Formal description of any Binary format file, please provide EBNF description, with dictionary etc., where each field has to be fully specified in terms of type and semantics of each possible value. Some examples have to be provided.

DE3.1.2A – Framework and Tools Specification (General and Model)

- Formal description of any language, rule based or functional or mix, by using EBNF description with dictionary and semantic description. Some examples have to be provided.
- Description of the high level communication interfaces such as COM, ACTIVEX, and support for plug-ins, etc., by providing: functions/procedures, functionalities, parameters, types of parameters, behavior, and internal behavior. In addition, a sample procedure and detailed sequence diagram of what can be done to exploit the module in other processes.
- Description of the User Interface, if any:
 - Visual Shape and design of the main frame
 - Menu with major and minor items and related associated functionalities.
 - Contextual menu
 - Main functionalities provided from the user interface
 - Visual Shape and design of the major dialog boxes.
 - Usage of tool bars, scrollbars, and any gadget or widget, etc.
 - Description of main activities of the users in terms of Use Cases, see the other deliverable
 - For usability aspects please consult ACIT partner
- In designing/specifying the tools/modules please take into account the following general aspects:
 - Configuration management
 - Please verify if some your components can be produced customizing a component produced by other partners or can used by other partners in other tools
 - Interoperability on different platforms,
 - Print capability of the information manipulated
 - Protection aspects (registration, certification, operation control, access to certifier, DRM, etc.), please consult protection experts
 - Help to support the users,
 - Multilingual support of the user interface and of the help
 - Undo support that could be obtained with controlling all commands
 - workflow and cooperative work support to be integrated with the Workflow tools that will be selected for AXMEDIS, etc.
 - insert an About for citing, copyright, AXMEDIS projects and EC in a proper manner, as will be defined later.
 - Refer to used standards providing references and documents for the other partners. These documents will be made accessible to all via WEB.
 - Declare any library and tools that you are going to use and the license level/type/cost for that tools/libraries, etc. According to the CA you have to be very carefully in using:
 - PEK, it has to be authorized
 - Libraries that may enforce some constraints in the exploitability or portability
 - Any used element/library, etc. has to be approved
 - Any non approved element cannot be used.
 - Etc.
 - Installation capability, it has to be installable in a very easy manner
 - Manual support for technical and user point of views
 - Please remember that if the tool/module belongs to the AXMEDIS Framework as defined in the CA, it has to be provided in source code to be included into CVS connected to the AXMEDIS portal.

3 Implementation Guidelines

The AXMEDIS framework is an environment for integrating and validating the new enabling technologies and new knowledge invented with WP4. At the beginning, this task will produce a set of guidelines for creating software components. These software components will be created including research algorithms as described in WP4. The guidelines and most of the components will be publicly delivered to the whole community together with the components developed by partners in the past and listed in the next table.

AXMEDIS is an Open tool since:

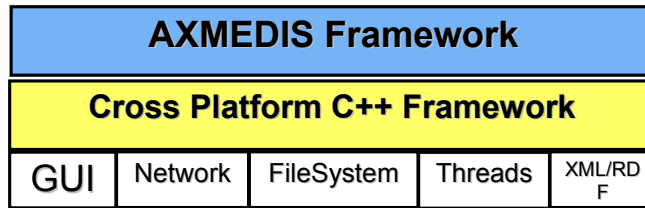
- The components of the framework will be available via the AXMEDIS portal in Open Source for the development or for transforming already present architecture in AXMEDIS compliant:
 - channel distributors, content providers,
 - viewers for the different platforms,
 - all the algorithms of the same type will be interchangeable, any innovation in the format, in the process, in the workflow, in the business model, in the DRMs, can be added into the framework without restructuring;
- The structure of the AXMEDIS framework will be formalised in terms of interfaces among its components. These will be mainly algorithms and software modules for managing content: composition, formatting protection, query, etc.;
- No limitation about the number of content providers and distributors and their access to the content and insertion into the AXEPTool network. Only certification will be needed and it will be provided almost automatically;
- Other distribution channels can be added according to the above solutions: direct channels such as those towards PCs or PDAs or, mediate via Channel Distributors such as those towards i-TVs;
- The same channel structure can be duplicated without any problem. You may have more Channel Distributors for i-TV, pay per view, etc. They can be set up for localization of content and services, for language and cultural differences, for providing content towards different technologies for providing different content;
- No limitation about the number of clients;
- No limitation about the number of transactions;

3.1 Decisions on Design, Implementation and Specification Tools (ALL)

- UML: a tool will be selected, VISIO 2003 professional is one of the candidates, other possibilities are UML free tools, etc. They have to be capable of processing UML, verification of consistency and managing packages, action diagrams, IDL, etc....
- Model for API is based on IDL, which in turn is based on UML formalisation
- All the tools including the AXMEDIS Object Manager have to be developed in C++
- AXCS and AXMEDIS object database have to provide for the database access some ODBC or JDBC, this has still to be decided
- Technology for AXPMS has not been fixed yet
- Communications with AXCS can be in SSL
- Database technology has to be scalable: SQL, MYSQL, DB2, Postgress, etc.
- AXEPTool has to be based on non proprietary protocol such as HTTP
 - The same DB used in the AXMEDIS database has to be used in the AXEPTool
- AXMEDIS content processing and editing tools have to be realized in C++, firstly for WINDOWS but taking into account of porting them on MAC/LINUX. So that a GUI abstraction is needed such as Mozilla or WXwin, a scripting tool for the GUI can be taken into consideration..

3.2 Identification of a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework

In this section a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework is identified.



The work performed is sketched in the following steps:

1. Identification of requirements for the C++ Framework
2. Selection of candidates
3. Evaluation of candidates

3.2.1 Requirements for the C++ Framework

The requirements identified for the C++ Framework to be used as the basis of AXMEDIS Framework are the following.

The **C++ Framework** has to:

1. be multiplatform supporting at least: MS Windows (98?, 2000, XP), MACOS X, Linux
2. NOT be under GPL license
3. NOT be under commercial license (NOT MANDATORY)
4. be rather wide spread
5. be usable from a C++ Application
6. allow the realization of standard user interfaces as well as MDI, drag&drop support, tree view, details view, clip board,...
7. allow run-time generation of user interfaces
8. allow the realization of custom views, where the application can control the visualization and the interaction with the view.
9. allow cross platform access to basic resources like: File system, Network (TCP & UDP), Threads, XML Parsing, RDF.
10. render multimedia information (video, audio, images) (RELEVANT)
11. allow the realization of applications with skin (like Windows Media Player) (OPTIONAL) or not prevent its realization on some platforms (MANDATORY).

3.2.2 Candidates for the C++ Framework

Possible candidates are:

- wxWidgets <http://www.wxwidgets.org>
- Qt <http://www.trolltech.com/products/qt>
- Mozilla XPToolkit <http://www.mozilla.org/xpfe/>
- tcl/tk <http://www.tcl.tk/>
- wxPython
- GTK
- MFC
- fox toolkit <http://www.fox-toolkit.org/>
- FLTK <http://www.fltk.org>
- PLIB <http://plib.sourceforge.net/>

(for a list see <http://home.pacbell.net/atai/guitool/>)

3.2.3 Evaluation of Candidates

For the candidates will be evaluated:

- platforms supported
- licence
- diffusion
- GUI capabilities
 - basic user interfaces
 - advanced user interfaces
 - support for automatic run-time generation of user interfaces
 - custom user interfaces
 - support for multimedia
 - skin support
- Platform abstraction
 - file system
 - network
 - threads
- Tools
 - xml support
 - rdf support

wxWidgets

General	
Platforms	Win32, MAC OSX, Linux, OS2, palmOS, winCE
Licence	LGPL
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible but not directly supported (XRC is used for XML description of GUI)
Multimedia Support	Native support for images (jpg, png, tiff), contributions are present for some multimedia support
Skin support	Feasible but not directly supported
Platform abstraction	
File system	Ok, it supports also zip files
Network	Ok (not UDP)
Thread	Ok
XML	Marginal, a non validating parser is present
RDF	No
Notes	
A library for XML parsing may be used (XERCES)	

QT

General	
Platforms	Win32, MAC OSX, Linux, Linux PDA
Licence	Commercial for Win32, GPL for other platforms
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible but not directly supported (?)
Multimedia	?

Support	
Skin support	Feasible but not directly supported
Platform abstraction	
File system	Ok
Network	Ok
Thread	Ok
XML	No
RDF	No
Notes	

Mozilla XPToolkit

General	
Platforms	Win32, MAC OSX, Linux, ...
Licence	LGPL
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Poor
Automatic GUI	Feasible, XUL helps in this task
Multimedia Support	?
Skin support	No
Platform abstraction	
File system	Ok
Network	Ok
Thread	Ok
XML	Ok
RDF	Ok
Notes	

tcl/tk

General	
Platforms	Win32, MAC OSX, Linux, ...
Licence	LGPL
Diffusion	Avarage
GUI capabilities	
Basic GUI	Good
Advanced GUI	Avarage (some user interaction is not “standard”)
Custom GUI	?
Automatic GUI	Feasible but not directly supported
Multimedia Support	?
Skin support	?
Platform abstraction	
File system	Ok
Network	Ok?
Thread	Ok?
XML	?
RDF	?
Notes	

--

wxPython

General	
Platforms	Win32, MAC OSX, Linux, OS2,
Licence	LGPL
Diffusion	Good (mainly for python)
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible (supports XRC a XML GUI)
Multimedia Support	Contributions are present for some multimedia support (only on some platforms)
Skin support	?
Platform abstraction	
File system	Ok
Network	Ok
Thread	Ok
XML	Ok
RDF	Ok?
Notes	
Python code can be embedded in a C++ applications	

GTK+

General	
Platforms	Win32, Linux, MACOSX?
Licence	LGPL
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible
Multimedia Support	?
Skin support	?
Platform abstraction	
File system	Ok
Network	?
Thread	Ok
XML	No
RDF	No
Notes	
GTK mappings exist for Python, Ruby an many other languages	

MFC

General	
Platforms	Win32
Licence	Commercial
Diffusion	Very Good
GUI capabilities	
Basic GUI	Good

Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible
Multimedia Support	?
Skin support	Feasible
Platform abstraction	
File system	Ok
Network	Ok
Thread	Ok
XML	Ok
RDF	No
Notes	

Fox Toolkit

General	
Platforms	Win32, Linux, MACOSX (not native)
Licence	LGPL
Diffusion	Avarage
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible
Multimedia Support	?
Skin support	?
Platform abstraction	
File system	?
Network	?
Thread	?
XML	?
RDF	?
Notes	

FLTK

General	
Platforms	Win32, Linux, MACOSX
Licence	LGPL
Diffusion	Avarage
GUI capabilities	
Basic GUI	?
Advanced GUI	?
Custom GUI	?
Automatic GUI	?
Multimedia Support	?
Skin support	?
Platform abstraction	
File system	?
Network	?

Thread	?
XML	?
RDF	?
Notes	

PLIB

General	
Platforms	Win32, Linux, MACOSX
Licence	LGPL
Diffusion	Avarage
GUI capabilities	
Basic GUI	Low
Advanced GUI	Low
Custom GUI	Ok
Automatic GUI	Ok
Multimedia Support	?
Skin support	?
Platform abstraction	
File system	Ok
Network	Ok
Thread	Ok
XML	No
RDF	No
Notes	
Used for 3D Games development	

3.2.3.1 Conclusions

In the above evaluation of candidates, are marked in red/orange/yellow the features that are problematic. The features marked in red strongly discourage the adoption of the framework, the ones marker in orange suggest to avoid to use it.

It results that wxWidgets, wxPython, and GTK+ are the frameworks that could be adopted. However it seems that wxWidgets gives more coverage especially since it can be used also in PDAs.

3.3 Supported platforms (All Partners)

The components produced by the research activities will have to be compliant with the languages and platform identified in the specification and in particular:

- Content production tools, editors, publication tools, engines, protection tool editor, etc.:
 - Language: C++
 - OS: Windows XP and 2000, and viable as a second choice also for MACOS X
 - GUI: WxWidget (the new version of the WxWindows)
 - XML Parser: XERCES
 - WebServices/SOAP: gSoap
 - TCP/IP library: that of WxWidget
 - STL can be used but a particular attention has to be given on using only functionalities and data structures that are supported in the STL for PDA (Pocket PC 2003)
 - Avoiding the usage of functionalities that are Microsoft specific
- Protection Manager Support
 - Client: all features as above in C++
 - Domain Home, Domain Factory and Server: Java and C++
- AXEPTool:

- Publication and loading engines: as the editor area above
- C++ for the tools
- P2P virtual database: Java if needed
- Graphic User Interface: WxWidgets
- AXMEDIS Database and Query Support and AXCS:
 - Database technology: MySQL or Postgres
 - Technology for coding logic: Java
 - DBC: JDBC for the access
 - Operating system: Windows and Linux.
 - User Interface: JSP or PHP, probably better the JSP formalisation in classes and graphic design
 - Web Server: Apace, TOMCAT
- Scripting language:
 - Java Script (ECMA Script)
- Workflow Manager:
 - based on OpenFlow or BizTalk
 - Plug in of the workflow: the same as the content production area
- The usage of AXMEDIS Error Manager for all the tools in C++ that use an AXOM
- The usage of AXMEDIS Configuration Manager for all the tools in C++ that use an AXOM
- Fingerprint and Metadata/Descriptors extractors:
 - the same as the content production area

Other information is collected into the AXFW specification document.

3.4 Development frameworks

Development should be done using C/C++ for all code-sharing components, whereas all internal components may be developed using other languages with related tools.

Since the source code of all the tools and components will be accessible to all partners, the choice of a preferred development framework will not limit the possibility to use a different one for regular development and to build for the preferred development framework before upload or on less regular basis.

The following are the preferred development platforms for the project partners under windows:

Partner	Preferred Development Framework
DSI	MS Visual Studio .NET 2003
DIPITA	gcc/CygWin
COMVERSE	MS Visual Studio .NET 2003, MS Visual Studio 6 (Service Pack 5)
EPFL	MS Visual Studio .NET 2003
EUTELSAT	gcc/CygWin
FHGIGD	MS Visual Studio .NET 2003 (MS VS 6.0 and gcc+eclipse as secondary frameworks)
ILABS	MS Visual Studio .NET 2003
HP	
TISCALI	
FUPF	MS Visual Studio 6.0 or gcc (linux tools)
XIM	gcc or MS Visual Studio .NET 2003
CRS4	gcc
SEJER	gcc/CygWin
UNIVLEEDS	MS Visual Studio .NET 2003
IRC	MS Visual Studio .NET 2003
EXITECH	MS Visual Studio .NET 2003 or gcc

Thus the most preferred development framework is MS Visual Studio .NET 2003.

For Java development the choice of a uniform virtual machine will allow to reduce the possibility of integration problems. The chosen JVM is Sun 1.4.2. The migration to a new JVM (1.5.0) will be made all together.

The use of a specific Integrated Development Environment for Java development is not set, but it is mandatory the adoption of ANT for having an easier integration (some IDE use ANT as integrated building system). For testing the use of JUNIT is appreciated. For development of WebServices Java WSDP 1.5 has to be used.

3.5 User Interfaces (All Partners)

For the user interface in C++ applications wxWidgets should be used.

3.6 Communication (All Partners)

For the communications among modules in different processes/machines WebServices is the preferred technology to be used. However in some cases other solutions like XML-RPC or custom protocols could be used, for example for the interaction with existing tools or for performance reason.

For the interaction with modules in the same process static or dynamic libraries should be used. COM/ActiveX technology could be used to interact with existing components.

3.7 Information exchange (EXITECH, All Partners)

This section contains guidelines on the way in which information will be exchanged, through documents posted on the WEB AXMEDIS portal and by synchronizing source code and test cases updates

The main mode for information exchange between the AXMEDIS members is the use of document repository on the WEB PORTAL. Refer to the portal specification for the operative procedures.

The documents editing has to follow the rules defined by the AXMEDIS consortium. The templates to be followed for the document editing have to be posted on the WEB site into the management activity in a dedicated folder. New templates have to be created any time a new type of document has to be produced (i.e. slide template, deliverable template, reports and management reports, UML diagrams, etc.). The document responsible and the activity coordinators have to verify if the received/posted content comply with the template.

In case a new document type is needed, the template can be proposed to the project coordinator, who, if accept it, will post it on the portal.

It is strongly advised against sending documents in attach to the reflector

Also for the code editing a “template” can be created. The programming style template can be based on some standard guidelines as:

- C++ Coding Standard defined at <http://www.possibility.com/Cpp/CppCodingStandard.html>
- Mozilla Coding Style Guide at <http://www.mozilla.org/hacking/mozilla-style-guide.html>
- Code Conventions for the Java™ Programming Language defined by SUN at <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>
- Apache coding standards at <http://jakarta.apache.org/turbine/common/code-standards.html>

The standard guideline can be adapted for the project needs.

3.8 CVS repository (EXITECH)

A CVS have to be set up in order to allow the work on the same source code. The CVS system will be choose after the evaluation of:

- Reliability capabilities
- Transfer security
- Binary file management
- Rule definition capability

DE3.1.2A – Framework and Tools Specification (General and Model)

- Access rights managements capability
- Costs (several freeware are available)

The CVS system is used for two apparently unrelated purposes: record keeping and collaboration. The collaboration purpose is the most important one but also the record keeping has to be considered. It is needed because the user should have necessity to see or to bring back the situation to a given date.

The CVS repository is accessed by using a CVS client software. From the web portal a service for browsing the repository should be implemented.

In order to maximize the availability the repository and the CVS server have to be installed in a dedicated server with a high hard disk capacity and raid controller.

Several versioning systems are available and already used by the developers communities around the world. The most famous one is the CVS used in several important projects (i.e. SourceForge projects at: http://sourceforge.net/docman/display_doc.php?docid=14033&group_id=1). There are no reason to invent new systems or to develop specific application in this area. In follow a brief list of (free or GPL) versioning systems is presented:

- CVS (Concurrent Versions System) at <http://www.cvshome.org>
- RCS (Revision Control System) at <http://www.cs.purdue.edu/homes/trinkle/RCS/>
- PRCS (Project Revision Control System) at <http://www.xcf.berkeley.edu/~jmacd/prcs.html>
- AEGIS at <http://aegis.sourceforge.net/index.html>
- Microsoft Visual SourceSafe at: <http://msdn.microsoft.com/vstudio/previous/ssafe/>

Independent by the tool that will be chosen the versioning system has to allow the possibility for each developer to browse the source code (with the definition of the access rights), download it, update the local repositories or perform a commit on the central repository.

The repository has to be backup daily.

Since the CVS system should be installed in a different computer a web interface should be used for commit or update source code files.

The web portal has to provide instruction for the service users

- How to provide source code
- How to post and update files
- File header template (posted into the managements activity as stated in the information exchange paragraph)
- Compile/Install procedures (*readme* file, see before)
- Rules for integration of reused code from other projects or state or the art
- Etc.

The web interface can use some third party application allowing a continuous integration of the source code and giving the guarantee of “compilable” code on the repository.

The interface should allow to see the code tree status, the last commits, who commit a not compilable code, update to a dated record, etc.

A continuous build process framework run a daemon process (build loop) which will periodically check the source control tool for changes to the codebase, build it if necessary, and send out a notification regarding the status of the build.

Some already implemented solution can be:

- Mozilla Tinderbox at: <http://www.mozilla.org/tinderbox.html>

- CruiseControl and CruiseControl .net at <http://cruisecontrol.sourceforge.net/index.html>
- Anthill and AnthillPro at <http://www.urbancode.com/products/anthillpro/default.jsp>
- Integration Guard at <http://www.urbancode.com/products/anthillpro/default.jsp>
- AEGIS at <http://aegis.sourceforge.net/>
- Draco.net at <http://draconet.sourceforge.net/>
- DamageControl at <http://damagecontrol.codehaus.org/>
- BuildBot (Alpha version) at <http://sourceforge.net/projects/buildbot/>

3.9 Coding Guidelines

For Java source code use guidelines <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>

For C++ source code:

Source files

- Header files have to have **.h** extension.
- Implementation source files have to have **.cpp** extension.
- All file names have to be lowercase (e.g. *axobject.h*).
- A header file should normally contain the definition of one class, more than one class can be defined in one .h file if they are strongly dependent.
- When including files from directories use forward slash “/” and not back slash “\” (e.g. `#include “wx/dc.h”`)
- Don’t use tabs to indent code, use 4 spaces.
- Use 76 columns for coding

Naming conventions

- class names should begin with capital letter (e.g. *AxObject*)
- class attributes and methods should begin with lower case (e.g. *model*, *getModel()*)
- local variables and functions should begin with lower case;
- macro, enum values and constants should be in all capital letters with “_” as word separator (e.g. *READ_ONLY*)
- namespaces should be in all lower case letters
- typedefs names should end with Type suffix (e.g. `typedef ParamValueType<int> ParamIntType`)
- for accessors use get/set prefixes (e.g. *getModel()*, *setValue(x)*)
- for classes belonging to the AXMEDIS Framework that are exposed to the framework user use the “Ax” prefix (e.g. *AxObject*), for classes that are not exposed this is not mandatory.

Code formatting

- for brace placement follow one of the following rule, but use the same for the whole module
 - Place brace under and inline with keywords:


```
if (condition)           while (condition)
{                         {
    ...                   ...
}                         }
```
 - Traditional Unix policy of placing the initial brace on the same line as the keyword and the trailing brace inline on its own line with the keyword:


```
if (condition) {         while (condition) {
    ...                   ...
}                         }
```
- in case a line is too long split it leaving the operator at the end of the previous line and align the starting of the continuing line with the expression start, examples:


```
if (aaaaaaaaaa &&
    bbbbbbbbbb &&
    ccccccccc &&
    dddddddddd &&
    eeeeeeeee)
{
    ...
}
```

```

aFunction(aaaaaaaaaa,
          bbbbbbbbbb,
          ccccccccc,
          dddddddddd);

```

Class Header Template

```

/**
 * A one line description of the class.
 *
 * #include "XX.h" <BR>
 * -llib
 *
 * A longer description.
 *
 * @see something
 */

#ifndef XX_h
#define XX_h

// SYSTEM INCLUDES
//

// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

// FORWARD REFERENCES
//

class XX
{
public:
    // LIFECYCLE

    /**
     * Default constructor.
     */
    XX(void);

    /**
     * Copy constructor.
     *
     * @param from The value to copy to this object.
     */
    XX(const XX& from);

    /**
     * Destructor.
     */
    ~XX(void);

    // OPERATORS

    /**
     * Assignment operator.
     */

```

```

    * @param from The value to assign to this object.
    *
    * @return A reference to this object.
    */
    XX& operator=(XX& from);

    // OPERATIONS
    // ACCESS
    // INQUIRY

protected:
private:
};

// INLINE METHODS
//

// EXTERNAL REFERENCES
//

#endif // XX_h_

```

Class Implementation Template

```

// SYSTEM INCLUDES
//

// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

#include "XX.h" // class implemented

//////////////////// PUBLIC //////////////////////////////////////

//===== LIFECYCLE =====

XX::XX()
{
} // XX

XX::XX(const XX&)
{
} // XX

XX::~~XX()
{
} // ~XX

//===== OPERATORS =====

XX&
XX::operator=(XX&);
{
    return *this;
}

```

```

} // =

//===== OPERATIONS =====
//===== ACCESS =====
//===== INQUIRY =====
//////////////////// PROTECTED //////////////////////////////////////
//////////////////// PRIVATE //////////////////////////////////////

```

for what not explicitly stated try to follow the guidelines provided in:

<http://www.possibility.com/Cpp/CppCodingStandard.html>

3.10 DOC generation

Documentation of source code (C++ and Java) will be integrated in the code using JavaDoc comments style.

3.11 Adoption of Libraries

Libraries used in the project are:

Library and version	Area	Licence
wxWidget – 2.4.2	GUI	LGPL
FL C++ Lib contribution to wxWidgets Lib	GUI	LGPL
STC C++ based on Scintilla editor, contribution to the wxWidgets Lib	GUI	LGPL
wxImagick	Image processing	LGPL
Imagick	Image processing	LGPL
openssl	Protection	LGPL
cryptlib	Protection	GPL and standard commercial license
xerces-C++ - 2.6.0	XML parser	Apache Licence 2.0
gSoap	Web services	LGPL
CURL – 7.12.13	Crawler	BSD
easysoap	Crawler	LGPL
expat	Crawler	LGPL
libxml2	Crawler	LGPL
ODBC	Crawler	LGPL
PEAR Library	Database	LGPL
Xerces	Database	Apache Licence 2.0
Xalan	Database	Apache Licence 1.1
XPath	Database	
OpenFlow – 1.1	Workflow	GPL 2.0
Zope – 2.7.3	Workflow	ZPL 2.0
Phyton -2.3	Workflow	
XmlrpcLib	Workflow	
Cexpat	Workflow	
Microsoft ASP	Workflow	Microsoft licence
Microsoft .NET	Workflow	Microsoft licence
DirectX SDK	Players	
splay	Players	LGPL
faac	Players	LGPL
im1_dmif_mp4	Players	ISO
im1_dmif_trif	Players	ISO
im1_dmif_remote	Players	ISO

iml dmifclientfilter	Players	ISO
DOCFRAC	Fingerprints	LGPL
GNU ghostscript	Fingerprints	GPL
XPDF	Fingerprints	GPL
HTMLDOC	Fingerprints	GPL
WordNet (Emglish, Italian, Spanish, French, German)	Fingerprints	Free for English, proprietary for other languages
TreeTagger	Fingerprints	Free for research. Proprietary for commercial use.
CLAM (0.7)	Fingerprints	GPL
Torch3	Fingerprints	BSD
LibSVM – 2.71	Fingerprints	LGPL
Libsndfile – 1.0.11	Fingerprints	LGPL
BeeCrypt	Fingerprints	LGPL
Botan	Fingerprints	BSD-style
CryptLib (?)	Fingerprints	GPL and standard commercial license
RtAudio – 3.0	Fingerprints	BSD-style open source
PortAudio – 18	Fingerprints	BSD-style open source
Libsndfile – 1.0.11	Fingerprints	LGPL
FFTW – 3.0.1	Fingerprints	GPL and Non-free license (see http://web.mit.edu/tlo/www/)
FFMPEG	Fingerprints	LGPL
FOBS	Fingerprints	LGPL
SpiderMonkey JavaScript Engine ver. 1.5 by Mozilla	Engine	LGPL
SoundTouch	Adaptation	LGPL

3.11.1 Licensing terms (All Partners)

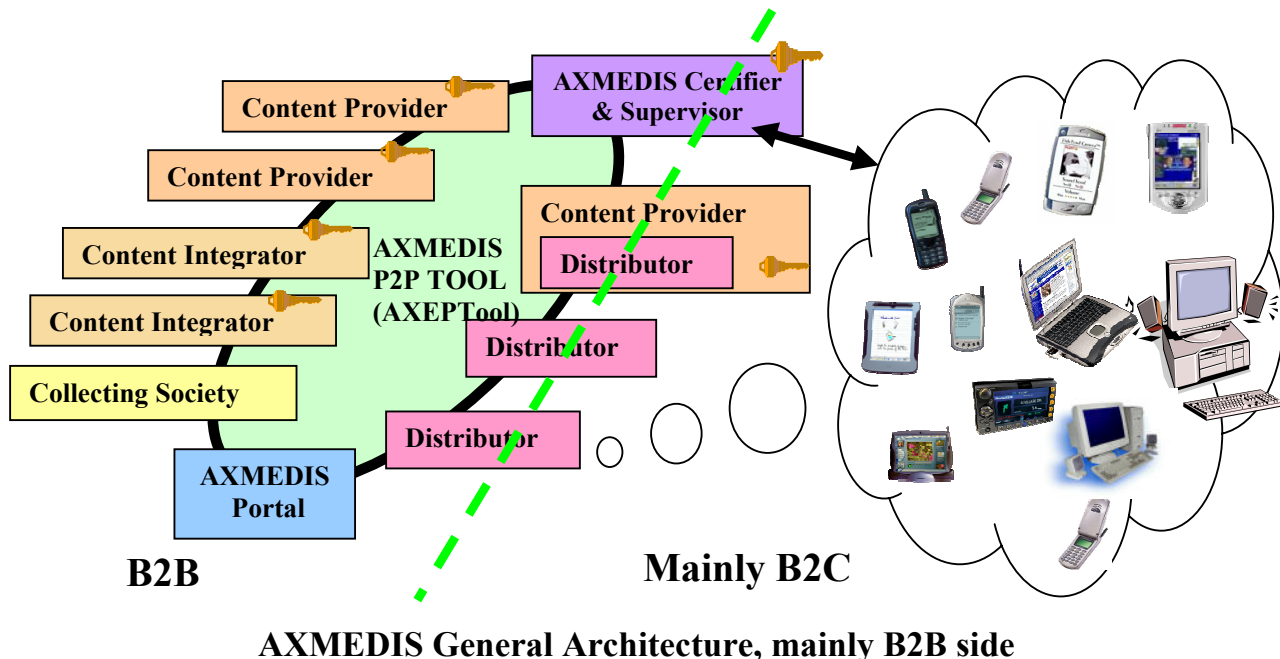
- license for libraries and access to code modality as stated in the CA for each module produced and reused

4 AXMEDIS General Architecture

The AXMEDIS digital content and content components (in the following, AXMEDIS content in general) will have a specific format capable of integration inside any kind of cross media format (video, images, animations, document, audio, etc.), adding metadata, identification, classification, categorization, indexing, descriptors, annotation, relationships and play activities and protection aspects. The format will permit the combination of content components, their secure distribution, etc., in the respect of the copyright laws, supporting a large variety of DRM rules and models according to concepts of interoperability among DRMs (mainly, but not only, based on MPEG-21, with both binary and XML low level formats). Within the AXMEDIS content any type of cross media content can be included from simple multimedia files to games, software components, for leisure and entertainment, infotainment, etc.

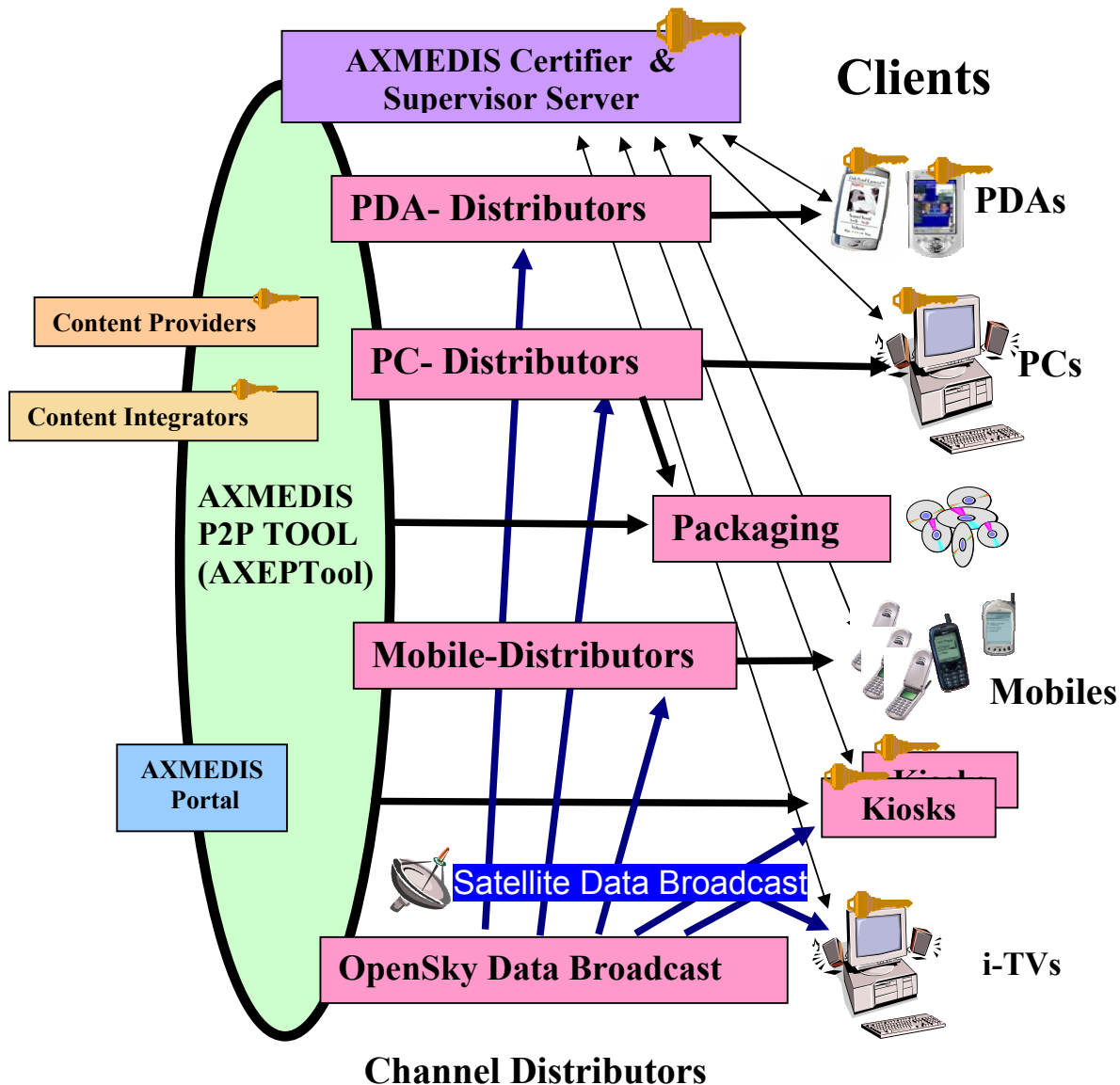
The General Architecture of AXMEDIS is represented in the next figure, which highlights both:

- **production** of AXMEDIS digital content and content components in connection with the AXMEDIS P2P tool (AXEPTool) that follows business mechanisms of B2B and support DRM with a certification authority (AXMEDIS Certifier and Supervisor). This can be connected to the Collection Societies as well as to each Content Provider and to Distributors if needed;
- **distribution** of AXMEDIS digital content towards clients via specific distributors that realize the last level of the distribution chain. This last level can also support a B2B transaction if the distribution is targeted at institutions. Also at this level the sharing via mechanisms of P2P is allowed and stimulated. This will not invalidate the protection model of AXMEDIS DRM.



The standard distribution channel is today a single distribution path for each type of content, and often, multiple proprietary systems of representation for the same content. The definition of distribution channel editorial formats would provide one way, unified and rock-solid content format for multipurpose applications. Alternative solutions support multichannel distribution by using an XML model of content into the Content management systems of the content provider that also include multiple transcoding engines for transforming the XML model of content into the format suitable for the channel. This approach is not flexible enough since the transcoding of content at the source strictly limits the management of Digital Rights. In fact, in models such as CONTESSA the DRM can be applied only to the content in its final version. This creates key problems for the content providers since the content distributors are entitled to receive unprotected content. This is almost unacceptable in most cases.

In AXMEDIS, the channel distributors may maintain their distribution process. They can continue to use the same format for reaching the final users. In AXMEDIS, the content is distributed by using the P2P tool, namely AXEPTool, by using an evolution of the MPEG-21 format, with the AXMEDIS contribution. This content will easily contain and deliver MPEG-4, MPEG formats, PDF, HTML, SVG, images, documents, videos, audio file, etc. (in open standard format for continuation, without the use of proprietary technologies) on demand and for all platforms according to the final format produced by the Distributor. The received content will be formatted by using AXMEDIS tools on the basis of specific editorial formats.



AXMEDIS General Architecture, mainly B2C

The possible Channel Distributors have a large variety of capabilities, they are both of pull and push, and may include off-line and on-line connection from the client to the distributor.

Channel Distributors are interested in:

- Getting AXMEDIS content and components from the Content Providers and using them for distributing content via their channels for redistribution for both B2B and B2C transactions.

- Collecting AXMEDIS contents in a local database for preparing the production content Programme that is the agenda/menu proposed to the customers and final users.
- Using AXMEDIS content for creating attractive content for their customers. For this reason, they need to have the possibility of inspecting content in their internal LAN on a client PC.
- Receiving and satisfying requests from their customers for delivering to them the proposed content
- Receiving and satisfying queries performed by their customers that are looking for specific content. This activity is one of the most interesting added value of the AXMEDIS architecture.
- Getting updated information about the possible content that can be recovered from all Content Providers. This activity is performed via a service of the AXMEDIS portal. The updating of the database of the available content is performed in push via satellite data broadcast with specific policies.
- Accessing statistics produced by the AXMEDIS Certifier and Supervisor about the content usage.

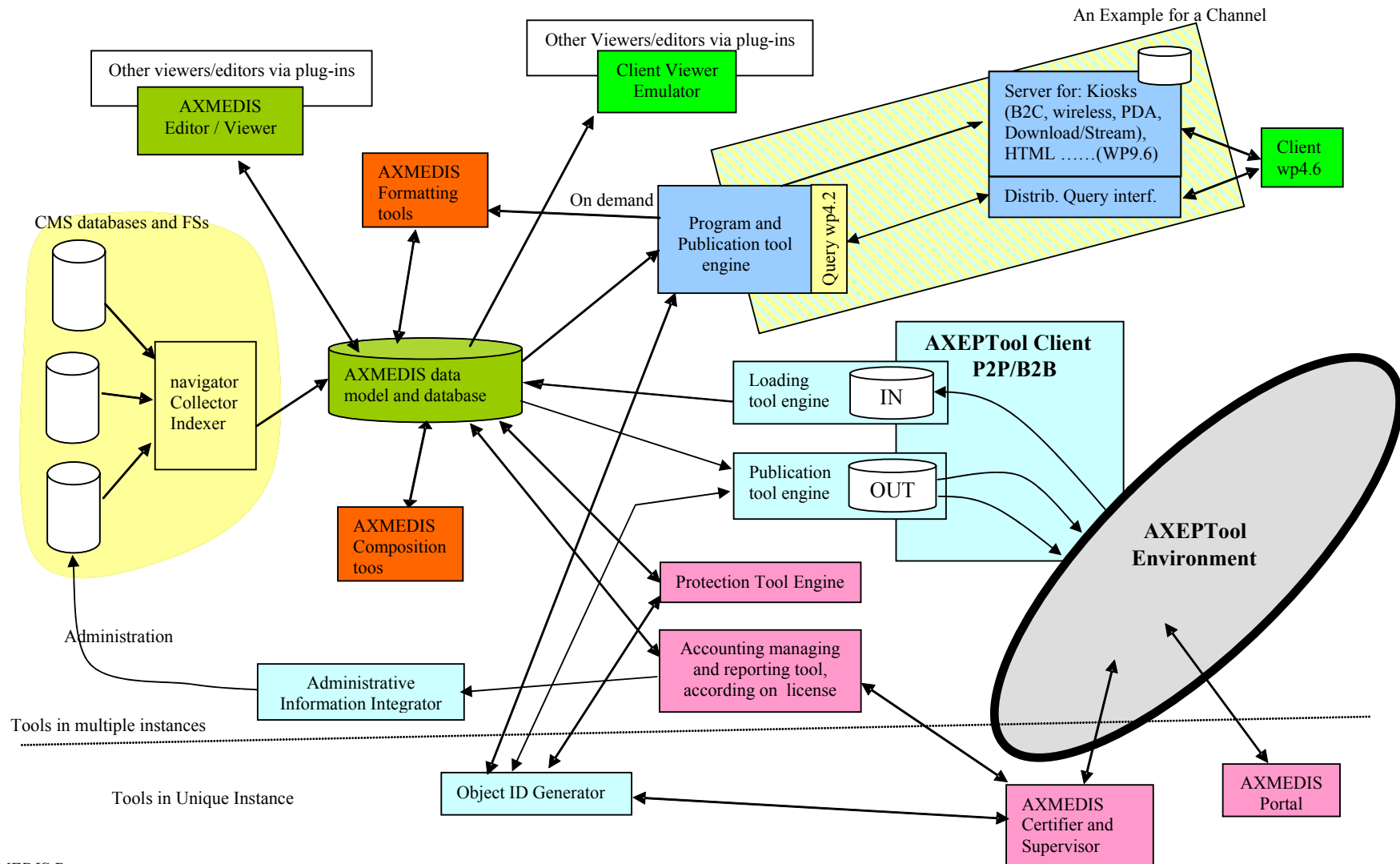
Satellite Data Broadcast It is a content distribution mechanism that permits the distribution of the AXMEDIS content in a very efficient manner. This improves the quality of service of the data delivery process (dependent on broadband availability in client location), and Distributors and also PC users can also rely on Satellite Broadcast. This technology, provided by EUTELSAT's Opensky platform, allows large quantities of data to be pushed via satellite directly on the user's PC without congesting local networks. The use of this technology is completely transparent with regard to the AXMEDIS process and only acts as a cost effective and efficient transport mechanism. The same technology also allows the content providers to bring live multimedia streaming content directly to the user's PC either for free to air content (mainly for marketing purposes) or paying on-demand channels. The pushing mechanism can be used to renovate the catalogue of the Distributors periodically at low cost.

This platform appears to be ideally suited for distributing AXMEDIS content and components. It represents an excellent opportunity for content providers for new business and for accelerating the distribution decreasing their costs.

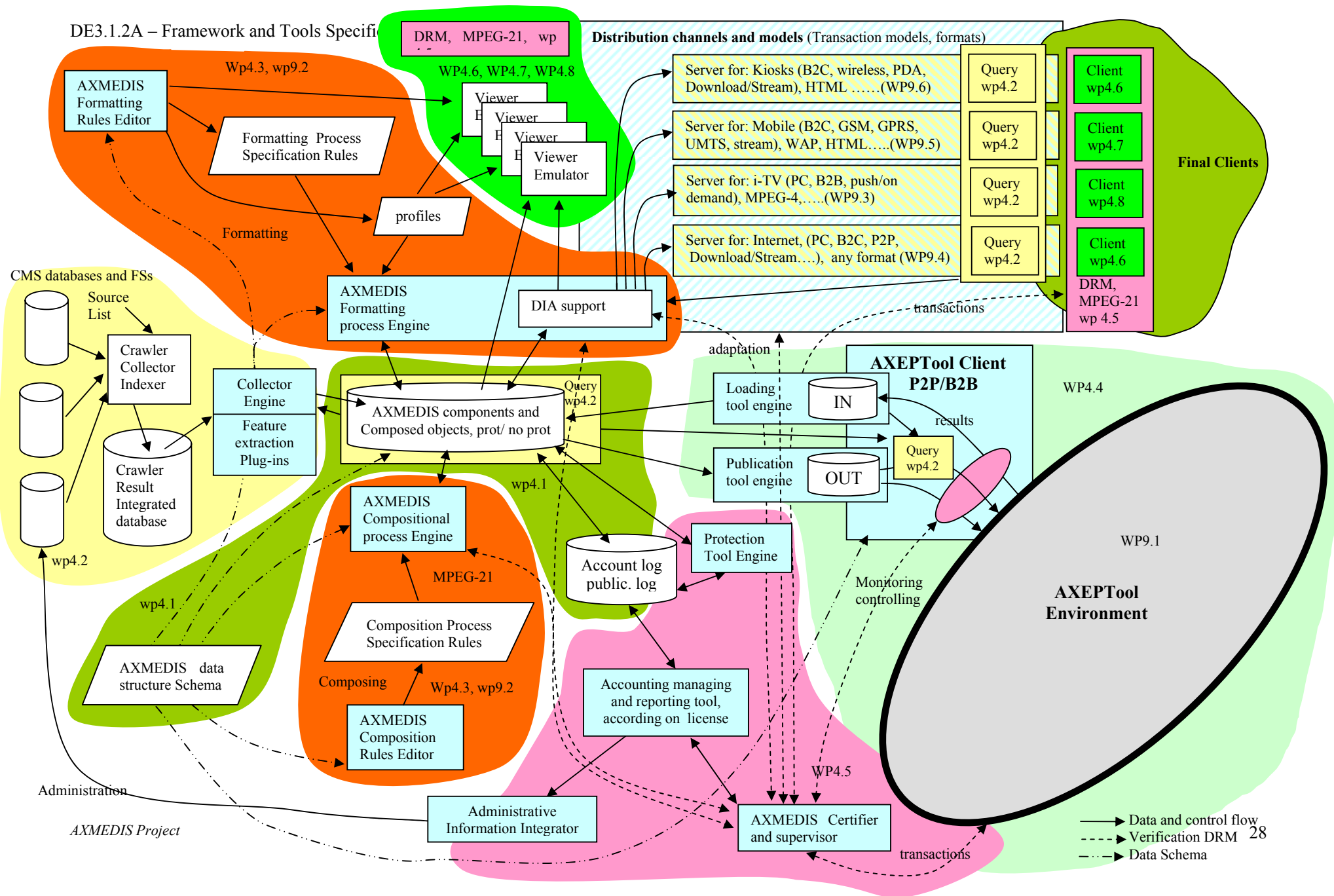
The satellite distribution channel can be used for several activities of content distribution for both B2B and B2C business models:

- The push of content
 - updating the AXMEDIS content and components in the databases of the Distributors and of the Providers;
 - updating the general indexing databases of the Distributors with updated information regarding the available AXMEDIS content and components of the Providers;
 - updating the AXMEDIS content on Kiosks;
 - delivering AXMEDIS content on demand directly to the consumers connected to the satellite i-TV according to their interactive requests;
 - delivering AXMEDIS content to the consumers connected to the satellite i-TV-PC according to their selection performed from the programmed content of the day and week.
- The streaming of AXMEDIS content on MPEG-4 on one or more channels for:
 - promoting Content Providers' content;
 - promoting Distributors' services, for example stimulating the acquisition of content in push with a business model based on subscription or pay per view;
 - Creating specific B2B channel with large institutions and consumers.

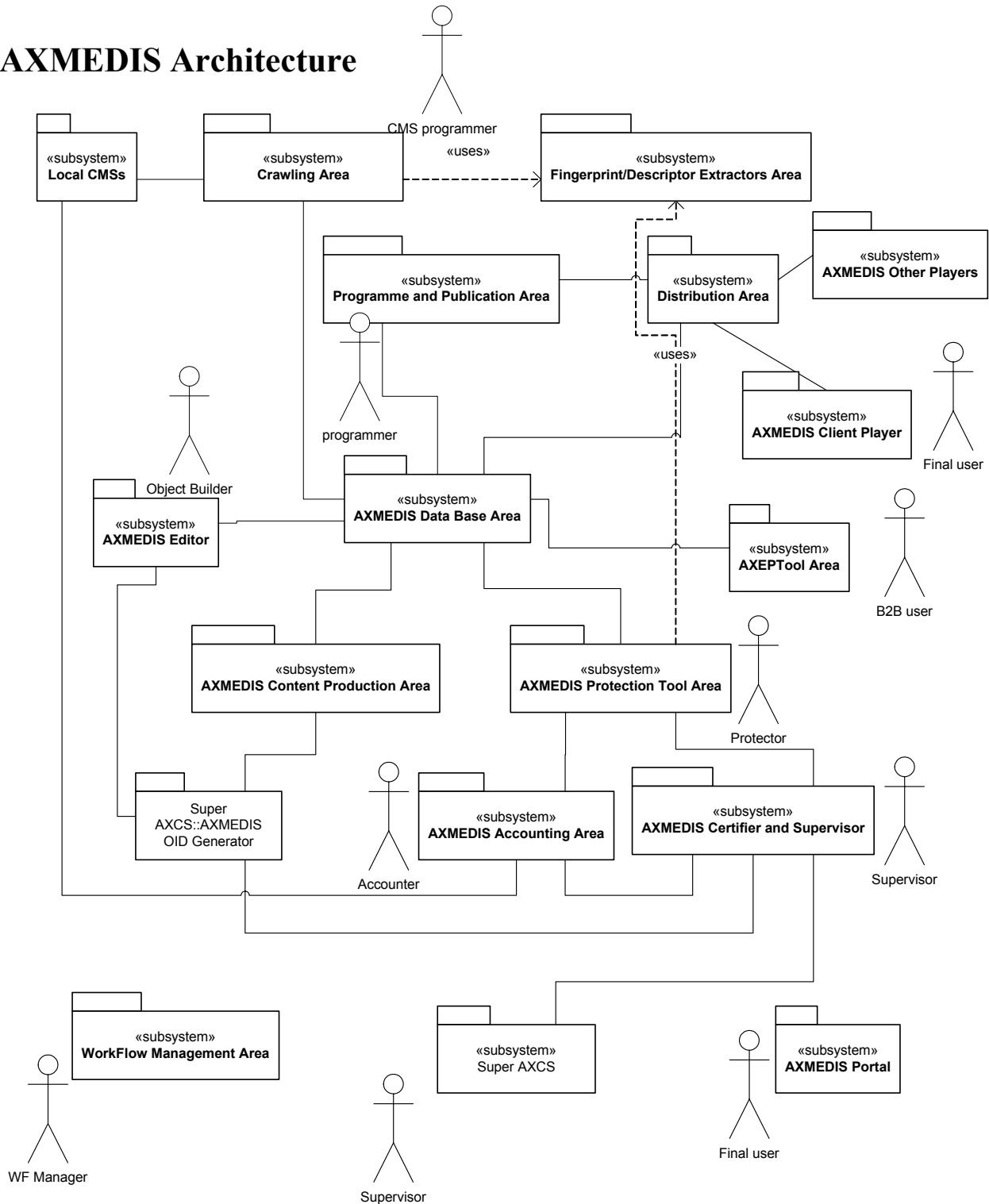
4.1.1 AXMEDIS General Architecture



DE3.1.2A – Framework and Tools Specific



AXMEDIS Architecture



AXMEDIS Framework is composed by the following areas and components:

- **Local CMSs** – represent the set of CMSs which are of interest to be connected to AXMEDIS system via Crawling Area tools.
- **Crawling Area** – includes Focuseek adaptation, Collector Engine, etc... includes a set of plug-ins which allows interaction among proprietary CMSs and AXMEDIS system to migrate digital resources towards AXMEDIS factory
- **Fingerprint Extractors Area** – includes access to a set of algorithms capable of estimating fingerprints (for collection of metadata and for certification) of a large variety of possible resource/content type for extracting a large amount of features. Such algorithms should be designed as a set of plug-ins usable in different AXMEDIS tools.
- **Programme and Publication Area** – includes all the components which allow the interconnection among AXMEDIS networks and databases to the distribution channels for producing programs to public content on the distribution channel. It also allows the management of request for content production/adaptation on demand.
- **Distribution Area** – includes tools to allow content distribution on specific channels and terminals.
- **AXMEDIS Client Players** – includes a reduced version of AXMEDIS Editor only capable to read and show AXMEDIS objects. This Area also includes all the plug-in versions of this tool which allow AXMEDIS object visualization in other players.
- **AXMEDIS Other Players** – includes all the plug-in versions of this tool which allow AXMEDIS object visualization in other players.
- **AXMEDIS Editor** – includes all the modules to create an AXMEDIS objects such as (i) a set of different views (ii) a command manager capable of validating user request (iii) a set of plug-in to use external resource/content editors/viewers (iv) a set of plug-ins to allow the integration of AXMEDIS Editor within other editing applications (v) interface with workflow, etc.
- **AXMEDIS Database Area** – includes AXMEDIS Database manger, access to the content, AXMEDIS Query Support, AXMEDIS Database, etc...
- **AXEPTool Area** – includes all components related to the B2B and P2P communication via AXEPTool application.
- **AXMEDIS Content Production Area** – includes all those tools needed for the automatic content production such as Formatting and Compositional Engine.
- **AXMEDIS Protection Tool Area** – includes DRM, encryption and decryption support, a support which allow communication towards clients and AXMEDIS OID Generator and Certifier and Supervisor, a tool for protecting objects, etc...
- **AXMEDIS OID Generator** – it is the unique responsible of object ID generation and it will elaborate all new-id requests coming from the client or automatic engines.
- **AXMEDIS Accounting Area** – includes a set of tools which allow final producers or distributors to collect information about what has been done on their objects in the AXMEDIS Circuit and thus to produce statistic analysis on content usage and other relevant statistical aspects. It also reports to the administrative side of the CMS the administrative information to prepare the bill at the content user.
- **AXMEDIS Certifier and Supervisor** – it is the responsible of user authentication and software certification, registration and tracking of the activities performed on AXMEDIS objects.
- **Workflow Management Area** – includes support and plug-ins to allow the content flow control, and programming, in all AXMEDIS subsystems.
- **Super AXCS** – is the unique responsible for supervising and collecting general information of monitoring for the actions performed on the content and for the registration of tools, users, etc.
- **AXMEDIS Portal** – includes services for partners, for managing the project, for working together, for supporting take up actions, for providing information during dissemination and demonstration.

All these tools and areas will be deeply analyzed in the following sections.

5 AXMEDIS Framework overview, General Architecture (DSI, all)

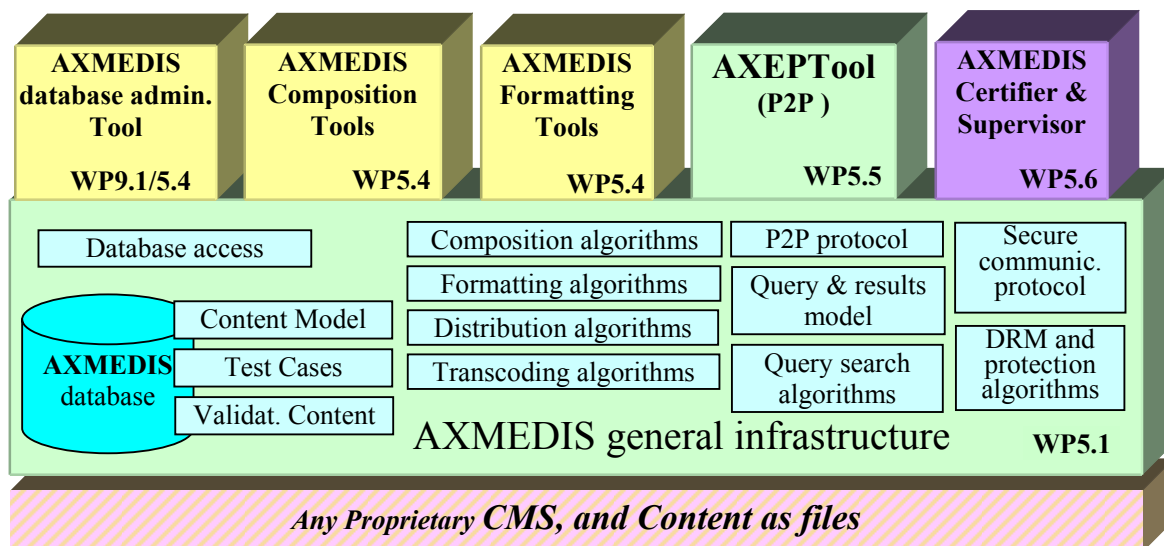
The added value of AXMEDIS will be in the research results producing innovative tools for content production (reducing costs and allowing production and formatting on demand) integrated in an open P2P tool (AXEPTool) at which any CMS could be joined for content production and distribution in the B2B environment of AXMEDIS. These aspects integrated with a set of sustainable demonstrators and take up actions will signal the age of real alliance in the content production process respecting the business of SMEs and permitting at them to access at relevant content and thus at producing content at a reasonable costs.

Most of the components mentioned in this WP are already available in the repository of the partners in an initial version and with less functionalities with respect to the real needs and the goals of the research activity planned for the AXMEDIS project. Some of the objectives have very ambitious objectives.

Some of the available components are in Open Source, supported by GPL or LPGL or other licenses, others are in the public domain, others still are proprietary. All the components provided by the partners will be made available to the community according to the rules specified in the Consortium Agreement in which each case is examined individually and carefully considered (the Consortium Agreement reports also a description of pre-existing knowledge and tools and the rules for their exploitation). A limited set of possibilities that range from LGPL, GPL, to proprietary modules have been used for labelling the IPR brought by the partners into the project. Some of these components will have to be customised to become compliant to the AXMEDIS framework, others will have to be transformed into multiplatform, only a few of them are to be built from scratch. Their maintenance along the project life is a mandatory activity to maintain their alignment to the AXMEDIS platform during its development. Partners that will provide proprietary modules have the duty to ensure they are AXMEDIS compliant.

The AXMEDIS Framework

In the following figure, the structure of the AXMEDIS Framework is reported. It contains all the necessary tools to manage the content workflow from the content production to the distribution over different channels, they are listed in the following.



The general infrastructure gives a common ground on the base of which other AXMEDIS-compliant applications can be built. The most relevant parts of the AXMEDIS **general infrastructure** are:

DE3.1.2A – Framework and Tools Specification (General and Model)

- **a common model for content** representation which allows that the information about the content can be exchanged between the different tools for the manipulation/fruition, a suitable solution could be the MPEG-21 format (see WP4.1).
- **Repository Administrator Tool** (including import/export, ingestion tools, from proprietary CMS solutions. This Module will be developed in WP9.1 for all the CMS of the partners involved in the project: SEJER, ILABS, WAN, XAURA, OD2, etc.).
- **access methods to the local AXMEDIS database** which stores the content with its metadata and it could be used for searches, classification, etc. as used in the AXEPTool and in any other content processing. All tools will use these access methods to get the object, write back the modification, create compound objects, etc. (see WP4.1)
- **AXEPTool and its P2P engine** for developing P2P applications allowing the content sharing, see WP5.5. P2P communication model, used by AXEPTool and available for other AXMEDIS compliant tools in the future.
- **Composition Tools** and algorithms that perform the composition of AXMEDIS objects, developed in WP5.4 as general tools and in WP9.1 with some more specific tools. Including AXMEDIS rules editor.
- **Formatting Tools** and algorithms that manage the presentation description of the AXMEDIS content, developed in WP5.4 as general tools and in WP9.1 with some more specific tools. Including AXMEDIS style editor for formatting.
- **all the transcoding and digital item adaptation algorithms** developed in WP4.3 and the other algorithms of the same type that will be developed in the demonstrators of WP9. Including multilingual management for metadata and for text, synchronisation tools and algorithms, etc.
- A protection model including:
 - **AXMEDIS Certifier & Supervisor** which controls the DRM and supervise the traffic on the AXEPTool.
 - DRM/Protection solutions, DRM engines, guidelines for licensing and contract definition, protection tools, monitoring tools, fingerprint estimation, enforcing and readers as developed in WP4.
- **test cases and validation content** for validating the identified algorithms and software components and tools included in the framework and for accepting new algorithms in WP5.2.
- **a format for query & results** to represent the request coming from the other peers about the shared content and the results that list the content corresponding the request criteria (see WP4.2). This has to be specified for simple client queries and for complex technical queries.
- **all the distribution algorithms** developed in WP4.6, WP4.7 and WP4.8. They are models for distributing content via I-TV, mobiles, PDA, PC, kiosk, etc.

The main idea of the AXMEDIS Framework

- Production of the AXMEDIS framework as a structured and organised set of software components in the area of cross media production, content sharing, content distribution towards multi-channel. These tools can be mainly used for the production process enabling content production on demand and automatic and semiautomatic production and formatting.
- Production of a general set of AXMEDIS production tools on the basis of the AXMEDIS framework.
- Improvement and maintenance of the AXMEDIS framework and tools.
- Preparation of guidelines to allow the development of the above mentioned points.

The main activities have been divided in the following subWPs:

- **WP5.1 -- Development of the general infrastructure**
- **WP5.2 -- Component Validation and Acceptance**
- **WP5.3 -- AXMEDIS framework integration and maintenance**
- **and all the other technical activities that are related to the tools specified in this document**

5.1.1.1 WP5.1 -- Development of the general infrastructure

This WP is coordinated by DSI

Period: M6-M30

The work in this WP includes several aspects and is quite distributed among all partners with a special attention to industrial partners such as COMVERSE, HP, ILABS, TISCALI that are interested at the exploitation of the produced results from the project.

T5.1.1: Production and maintenance of AXMEDIS guidelines

Managed by DSI with the work of all partners.

The algorithms developed in WP4 will have to provide a suitable interface to be joined each other to build the AXMEDIS framework designed and developed in this WP5. To this end, each software components including specific algorithms for content manipulation will have to be compliant to a specific interface. For example, the identified typical interfaces are those of the algorithms for (i) content transcoding, (ii) content composition, (iii) estimating a fingerprint, (iv) content formatting, (v) reading watermark, etc. These interfaces will be defined in this SubWP at which all the WP responsible of WP4 are called to work. Most of the algorithms have to share a common format and in some cases they have to be capable to communicate according to a unique data model. This data model will be based on the Object Oriented paradigm. The partners have a large experience in this sense. In order to reach these objectives, a set of guidelines will be produced according to the AXMEDIS specification to help the developers to work in precise manner. This will guarantee the interoperability of the research results obtained. This will be quite similar to what has been done in the past for other projects.

T5.1.2: Production of Stub Components of the general framework

Managed by EPFL. The work should be assigned on the basis of competencies to all the partners involved on this WP.

Based on the guidelines the second step will lead to the creation of the stubs of the major components and algorithms. Some of these stubs may become real modules (with common interfaces) in a short time by using the components provided by the partners and reported in the above table (i.e. transcoding algorithms). The creation of the stubs will help to avoid integration problems of the various components/algorithms in the framework in the later developments.

This process will start in the early months and continue through the project improving the interfaces of the components and their number in the AXMEDIS framework.

5.1.1.2 WP5.2 -- Component Validation and Acceptance

This WP is coordinated by FUPF

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. 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. 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.

5.1.1.3 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, 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.

AXMEDIS Editor



DE3.1.2A – Framework and Tools Specification (General and Model)

AXMEDIS Editor shall manipulate AXMEDIS object in respect of DRM which has been granted to the user on that object, e.g. a user could hold an object on which he hold play/view grants while he has not copy, move or other (manipulation) grants; in such a situation AXMEDIS Editor should stop every attempts of modifying the object by the user.

AXMEDIS Editor includes (or use) the following modules:

- **AXMEDIS Object Manager** – an AXMEDIS object model container wrapped for secure AXMEDIS object content manipulation. See Section 9;
- A set of viewer/editor for rendering or manipulating the information contained into the AXMEDIS Object Manager and model;
 - **AXMEDIS Object Editor and Viewer** – a user interface capable of “playing” AXMEDIS objects according to the structure and main MPEG21 controls. See Part B;
 - **DRM Editor and Viewer**, a view for editing and verifying the DRM rules for the users. See Part B
 - **Hierarchy Editor and Viewer** – a view for visualizing and modifying the structure of a document in terms of elements and their parent-child relationship. Hierarchy Editor/Viewer is the main entry-point to interact with the object and parts thereof, to recall other kind of editor/viewer, etc... See Part B;
 - **Metadata Editor and Viewer** - a view to edit and view metadata associated with the object. See Part B.
 - **Workflow Editor and Viewer** - a view to see the status of the object with respect to workflow management. See Part B.
 - **Behaviour Editor and Viewer** – a view for editing the behaviour of the object. See Part B.
 - **Visual Editor and Viewer** - a view for editing the visual rendering of the object. See Part B.
- **Protection Manager Support Client** – a set of functionalities to verify the protection and the rights (DRM), for the content contained into the AXMEDIS Object Manager. It contacts Protection Manager Support that in turn contacts AXCS, for certification, registration, accounting, etc. See Part H.
- **External Editor/Viewer Activation Manager** and relative external application plug-ins – gives, to AXMEDIS Editor, the capability of viewing/modifying resources by using external application which have not ActiveX/COM interface. It has a list of possible external applications which are compliant with AXMEDIS protection model; See Part B.
- **ActiveX Manager for Editor/Viewer** and relative ActiveX application plug-ins – gives, to AXMEDIS Editor, the capability of viewing/modifying resources by using external application which have ActiveX/COM interface; See Part B.
- **Internal AXMEDIS Resource Editor/Viewer** – a set of editor/viewer built-in AXMEDIS Editor which guarantees a range of basic behaviors, for example, audio player, video player, doc viewer, etc.; See Part B.
- **AXMEDIS Editor Configuration Manager** – is the responsible for configuration storage and maintaining of any AXMEDIS Editor or AXMEDIS client; See Part B.
- **AXMEDIS Content Tools Error Manager** – is a support for managing errors in the area of content processing, editing, formatting, etc. See Part B.
- **AXOM Content processing** -- This plug in interface allows to demand content processing to external algorithms. It has to allow the presentation of the some functionalities of the algorithms also to the user, e.g. fingerprint extractors, for Digital Item Adaptation, etc.; This interface for plug in is mainly usable for demanding content processing from out side. See Part B.
- **AXOM Commands and Reporting** – This plug in interface allows to control the action of the AXMEDIS Object Manager and to send messages and controls outside. See Part B.
- **Plug-in Manager** – allows the use of external plug-in which can be used for accomplishing various tasks (e.g., workflow plug-ins). These plug ins have to be certified in some manner to guarantee the safeness of their environment. The communication on these plug-in has to be performed in some protected manner since the content is going to be processed by them. See Part B.

Further AXMEDIS Editor specifications are certainly:

DE3.1.2A – Framework and Tools Specification (General and Model)

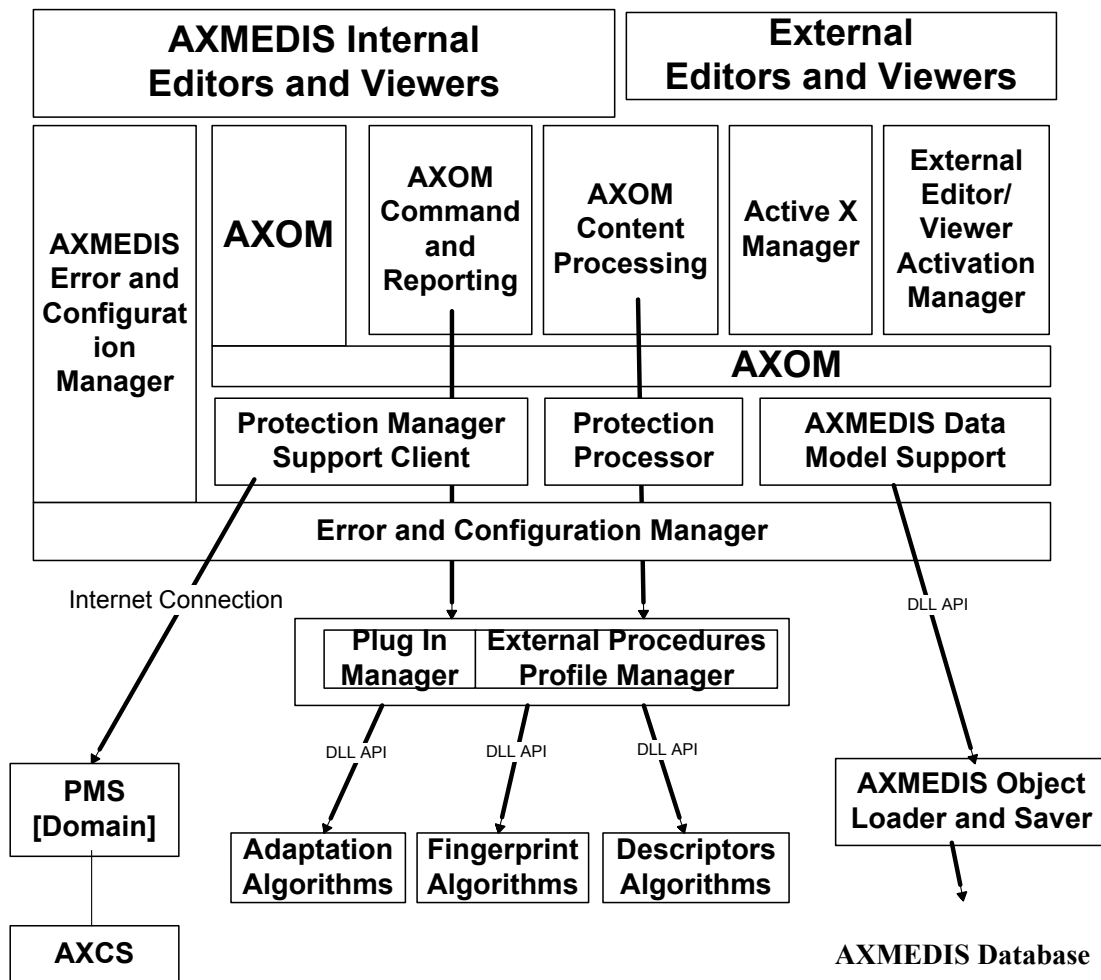
- Each configurable AXMEDIS Editor modules shall conform to AXMEDIS Editor Configuration Manager requirements. Moreover, each configurable AXMEDIS Editor modules shall own a default settings set in order to work also on AXMEDIS Editor Configuration Manager fault or absence;
- AXMEDIS Editor and all its sub-modules shall support multi-language. Multi-language option will be managed through AXMEDIS Editor Configuration Manager User Interface;

In this section will be analyzed only those modules which belong to AXMEDIS Editor area, the others will be analyzed in the respective sections.

Module Profile		
AXMEDIS Editor		
Executable or Library(Support)	Executable	
Single Thread or Multithread	Multi-Thread	
Language of Development	C++	
Responsible Name	Davide Rogai, Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
Config Files		Config Files Schema
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Views Manager	C++	wxWidgets
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
wxWidgets		

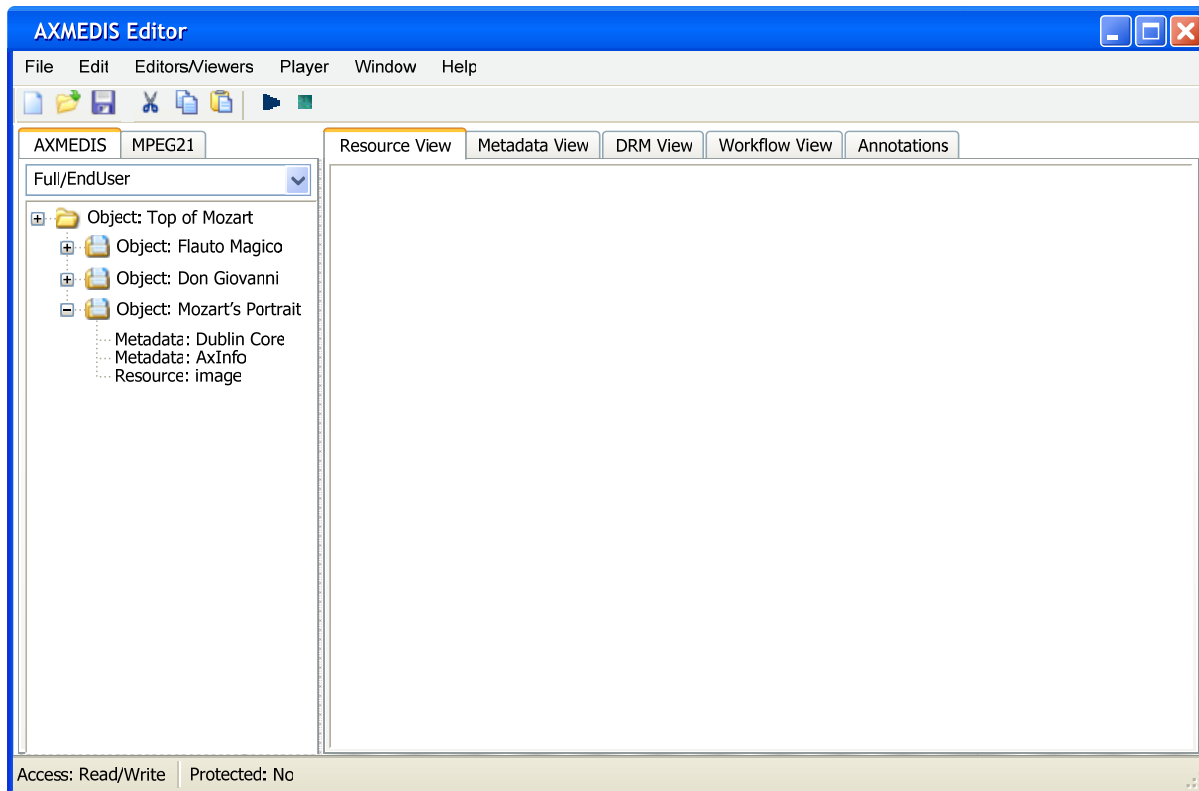
6.1 Software Architecture

AXMEDIS Editor Software Architecture



6.2 Graphic User Interface

The following is a possible user interface for the AXMEDIS Editor, a Frame is used for each object opened. Inside the frame, on the left a tree view representing the object structure is presented (see part B for details on the Hierarchy View) and on the right the different views of the object are hosted (see part B for the different viewers/editors). The AXMEDIS Hierarchy can be shown also as it will be visible from the End User by selecting the specific Combo Box.



AXMEDIS Editor

The top-level menus are:

File	Description
New	creates a new AXMEDIS object in a new window
Open...	loads an AXMEDIS object in a new window
Close	closes the current window
Save	saves the object to disk
Save as...	saves the object
Open from database...	opens an object from the database
Upload into database...	uploads the object into the database
Configuration...	opens the configuration editor
Exit	closes the application
Edit	Description
Undo	to undo the last operation
Redo	to redo the last operation undone
Cut	cuts the element
Copy	copy an element
Paste	paste an element

DE3.1.2A – Framework and Tools Specification (General and Model)

Delete	removes an element without putting it in the clipboard
Plugin...	Shows the plugins that can be used on the selected element
Editors/Viewers	Description
Resource Editor & Viewer	Opens the resource viewer
Metadata Editor & Viewer	Opens the metadata editor & viewer
Annotation Editor & Viewer	Opens the annotation editor & viewer
Visual Editor & Viewer	Opens the visual view on the whole object
Behaviour Editor & Viewer	Opens the behaviour editor & viewer on the whole object
DRM Editor & Viewer	Opens the DRM editor & viewer
Workflow Editor & Viewer	Opens the workflow editor & viewer
Player	Description
Start	Starts playing the object
Pause	Pause object palying
Stop	Stop playing
Window	Description
...	contains the list of windows currently opened
Help	Description
Guide to AXMEDIS Editor...	Opens the guide to the AXMEDIS Editor
About AXMEDIS Editor...	Opens a dialog showing information on the AXMEDIS Editor

Contextual menus are used in the AXMEDIS Hierarchy to perform specific operations on each element of the hierarchy:

Open	open a default view for the element
Open with...	in case more choices are available let the user choose
Properties...	show properties of the element
Cut	cuts the element
Copy	copies the element
Paste	pastes an element after the selected one
Delete	removes the element
Move up	move the element up
Move down	move the element down
Insert	
metadata...	inserts a new metadata element
resource...	inserts a new resource element
object...	inserts a new object
Plugin...	let the user select the operation to be performed on the element

On the MPEG21 Hierarchy specific contextual menus are used:

Open	open a default view for the element
Open with...	in case more choices are available let the user choose
Properties...	show properties of the element
Cut	cuts the element
Copy	copies the element
Paste	pastes an element after the selected one
Delete	removes the element without putting it in the clipboard
Move up	move the element up
Move down	move the element down

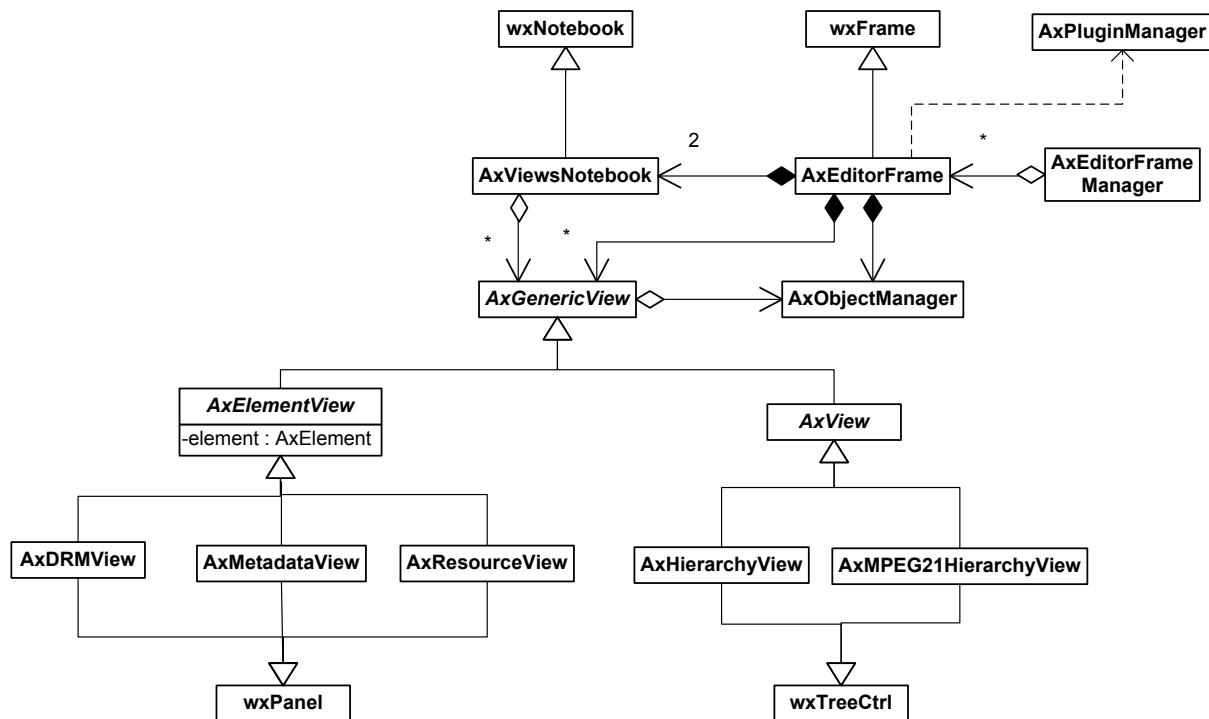
Insert	
Descriptor	inserts a new descriptor
Statement	inserts a new statement
Item	inserts a new item
Component	inserts a new component
Resource	inserts a new resource
Container	inserts a new container
Reference	inserts a new reference
...	
Plugin...	let the user select the operation to be performed on the element

Drag and drop can be used to copy an element from one hierarchy to another hierarchy (of different objects) or to move an element from one point to another (in the same object).

Files (images, video, documents, ...) can be dropped inside an object from the file system to become automatically resources belonging to the object.

6.3 Class Diagram

The following is the class diagram:



where class *AxEditorFrame* is the main frame, it contains:

- an *AxObjectManager* to manipulate the AXMEDIS object;
- a set of views currently open for the object;
- two *AxViewsNotebook* (one on the left and one on the right).

The *AxViewsNotebook* are used to contain different views of the object that can be hosted inside a Notebook, they can be: *AxDRMView*, *AxMetadataView*, *AxResourceView*, *AxHierarchyView* or *AxMPEG21HierarchyView*.

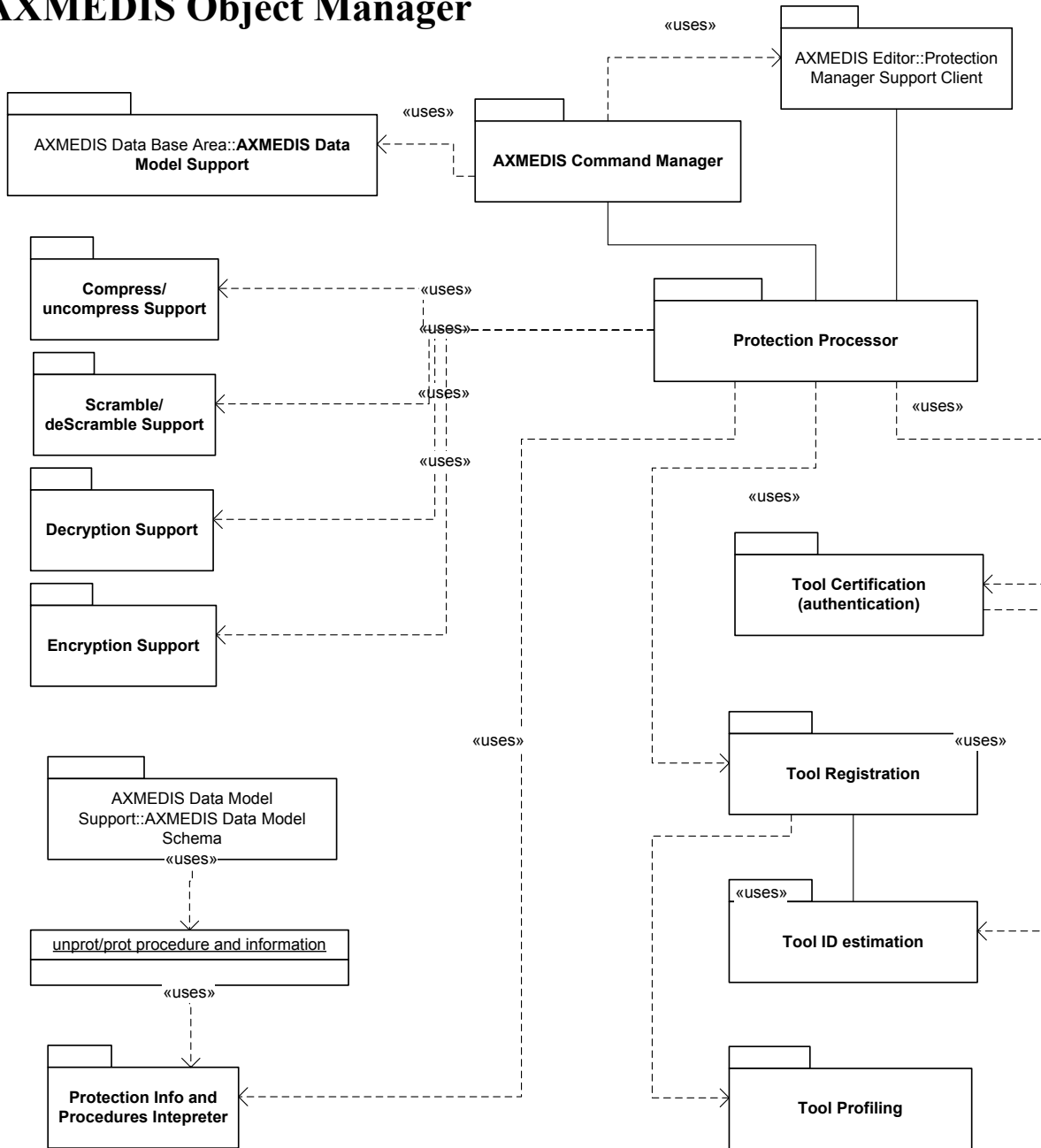
Abstract class *AxGenericView* represents a view on an *AxObjectManager*. *AxView* represents a view on the whole object (like a hierarchy view) and *AxElementView* is a view on a specific element of the object.

AxEditorFrameManager is a singleton class where each *AxEditorFrame* registers itself on creation. It allows to know which frames are currently open.

7 AXMEDIS Object Manager (DSI, EPFL)

AXMEDIS Object Manager, so called AXOM, is the coordinator of all other modules used by or built in AXMEDIS Editor. Coordination activities it is important to permit development of others AXMEDIS Editor modules almost independently each other. AXMEDIS Object Manager guarantees DRM rules respect on AXMEDIS object manipulations.

AXMEDIS Object Manager



The libraries for Dec-Enc, Scramb-deScramb, etc.. has to be built in a separable manner, so as to include all of them in the editor an player while in the player we have to have only de-scramble, decrypt, etc..

AXMEDIS Object Manager works in respect of AXMEDIS Data Model Support, reported in the following section; AXMEDIS Object Manager fundamentally is composed by three modules:

- **AXMEDIS Command Manager** which is the real interface for processing content models as the AXMEDIS Data Model and protected according to the tools of the AXMEDIS Protection processor.
- **AXMEDIS Data Model Support** to model the AXMEDIS objects according to the MPEG21 and additional requirements identified
- **AXMEDIS Protection Processor** to protect and processing registration, certification, and to protected and unprotect digital resources according to a dynamic mechanism similar to that of IPMP of MPEG21.

Module Profile		
AXMEDIS Object Manager		
Executable or Library(Support)	Library	
Single Thread or Multithread	Single Thread	
Language of Development	C++	
Responsible Name	Davide Rogai, Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

7.1 AXMEDIS Command Manager (DSI)

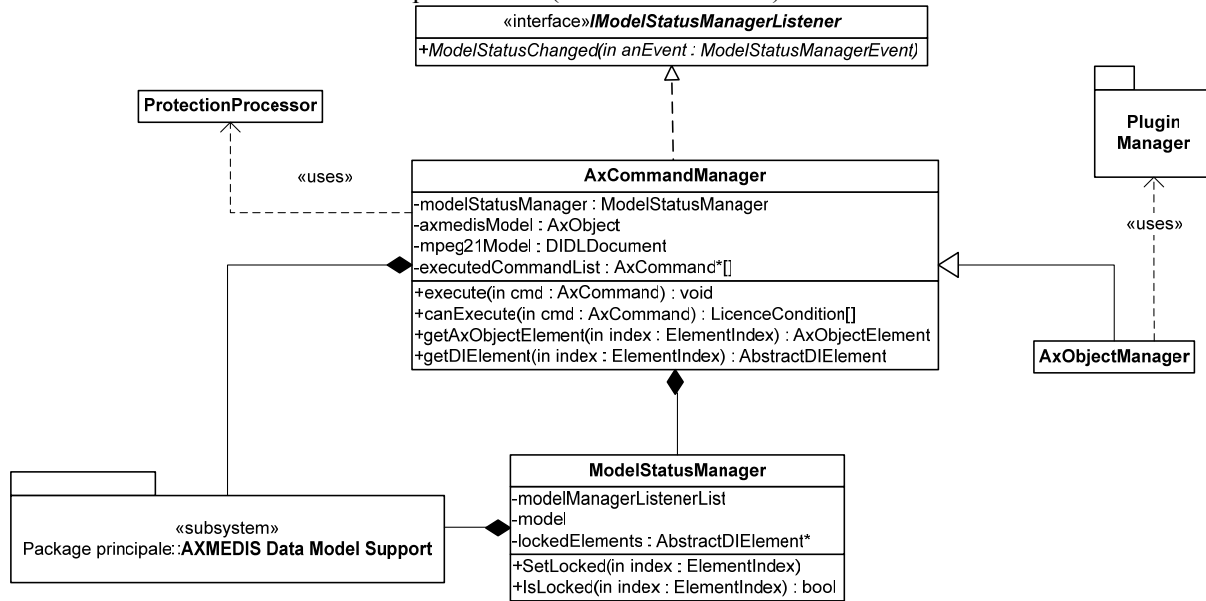
Module Profile		
AXMEDIS Command Manager		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Single Thread	
Language of Development	C/C++	
Responsible Name	Davide Rogai, Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

Command Manager is the interface among objects representation and all others data-manipulation AXMEDIS Editor modules (e.g. View Modules, plug-ins, etc...). Command Manager will provide all base operations (add, change, delete, etc...) which will be needed to manipulate AXMEDIS objects. Moreover, it will invoke Protection Manager Support to verify each operation.

- Command Manager works in respect of DRM model, i.e. on every user action it shall invoke the control of user grants on the involved items. That should be possible through the invocation of Protection Manager Support;
- Command Manager stores information about taken actions, in particular the following information shall be stored:
 - Kind of action and entities involved;
 - Who takes the action;
 - Where the action have been taken (AXMEDIS Editor installation identifier);
 - When the action have been taken (timestamp);

Command Manger provides an interface to permit development of data-manipulation plug-ins by third party developer. By data-manipulation plug-in is intended software component which can manipulate memory-stor

DE3.1.2A – Framework and Tools Specification (General and Model)



Fundamentals classes involved in the Command Manager package are:

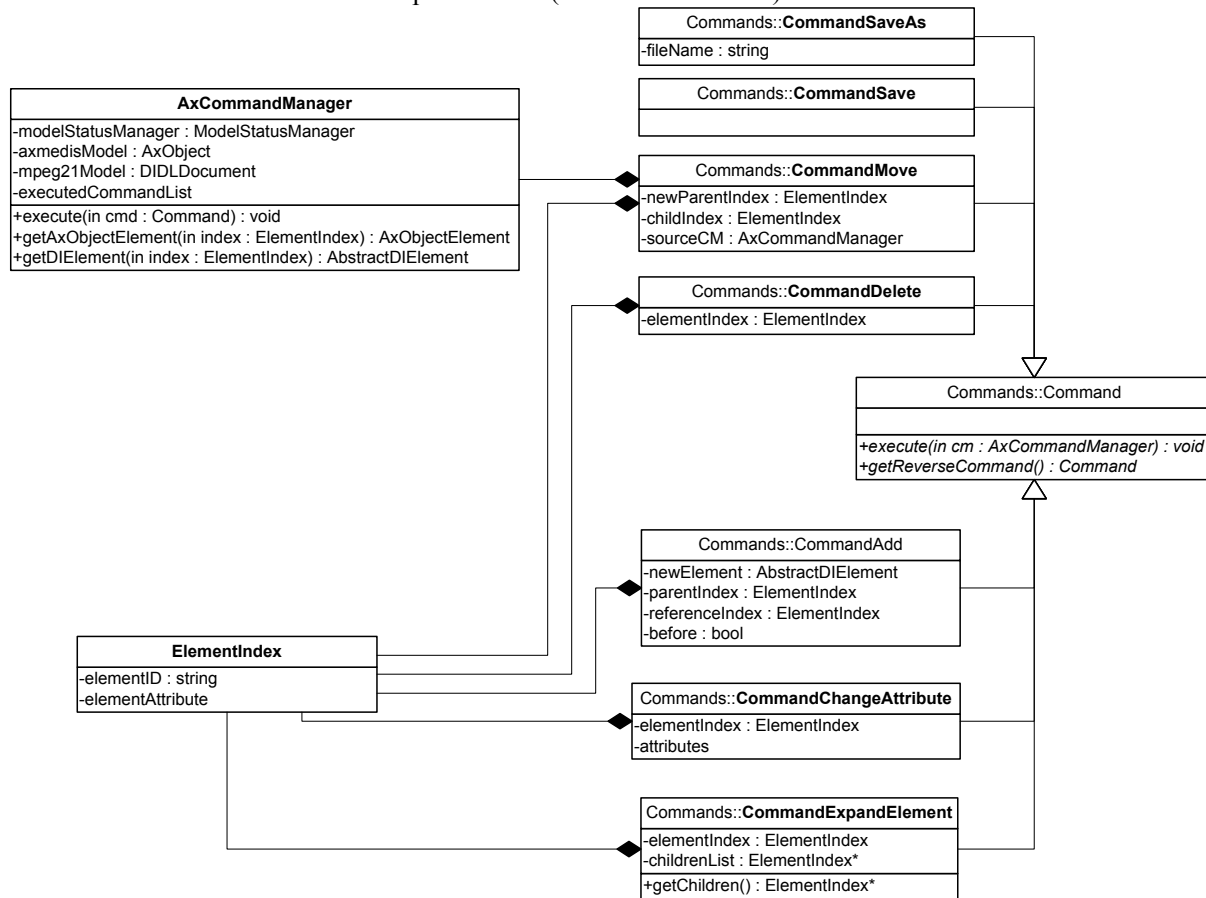
AxCommandManager: it processes all the requests of command execution exposing the execute command method. It owns a reference to a data model (model) (from Data Model Support) that manage the real document and it is used by the commands to perform their operations. Singleton class ProtectionProcessor is used to delegate all the security checks, i.e. certification and verification of the device.

It implements the **IModelStatusManagerListener** interface to receive notification of model status changes.

ModelStatusManager: it manages all the information needed to maintain the consistency of model data: it prevents concurrent write/read to elements of the document.

AxObjectManager: derived from AxCommandManager, it will be the interface to the module, it will allow also to access to PluginManager functionalities.

DE3.1.2A – Framework and Tools Specification (General and Model)



Commands on the object are generalized in an abstract class **Command** which exposes:

- execute method in order to be flexible to the introduction of new commands and
- getReverseCommand to support undo mechanism

Some implementation examples are provided:

- **CommandAdd**: it allows to add an element to the document, it requires the new element and the location of this addition
- **CommandChangeAttributes**: it allows to modify specific attributes of an element, it requires an attributes list and information to get the position of the element in a document
- **CommandMove**: it allows to move elements within a document and between two distinct document, i.e. it allow to realize cut-and-paste operations. It requires references to the node to be moved, to the new position of the element and to the model or **CommandManager** the element comes from
- **CommandDelete**: it allows to delete an element of an existing document, it obviously requires a references to the element to be deleted
- **CommandSave**: it allows to save the modified document in the location the original document comes from, i.e. it override the old version of the document with the modified one
- **CommandSaveAs**: it allows to save the document (modified or not) to a new location. The command requires to know the new location for the document, e.g. an URI or an ODBC database name, etc...

All these commands use the **ElementIndex** class which is the unique type of reference to elements which the user of the **AxCommandManager** class can use to manage the model of the document. **ElementIndex** is an indirection layer between the final user of the document model (i.e. who creates tools to manage AXMEDIS objects) and the model itself. In that way, controlled access to information and content contained in a document is guaranteed.

7.2 Protection Processor

Module Profile		
Protection Processor		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	C/C++	
Responsible Name	Andrea Vallotti, Davide Rogai	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
Certificate X.509v3		RFC 2459 http://www.ietf.org/rfc/rfc2459.txt RFC 3280 http://www.faqs.org/rfcs/rfc3280.html
PKCS #12 Personal Information Exchange Syntax Standard		PKCS #12 http://www.rsasecurity.com/rsalabs/
PKCS #7 Cryptographic message syntax standard		PKCS #7 http://www.rsasecurity.com/rsalabs/
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
cryptlib	Cryptlib 3.1	cryptlib is distributed under a dual license that allows free, open-source use under a GPL-like license and closed-source use under a standard commercial license. In addition, cryptlib is free for use in low-cost, non-open-source applications such as shareware, and for personal and research use.
Xerces-C++	Version 2.6.0	Apache License, Version 2.0
Apache XML Security	Version 1.1	Apache License, Version 2.0

Protection Processor has mainly four tasks:

1. To register and certify an AXMEDIS tool containing the AXOM, e.g. editor, player, engine, etc...
2. To control software which uses sensible content and does not contain AXOM, e.g. plug-ins for fingerprint
3. To reveal attacks during tool execution, e.g. code debugging

4. To protect and un-protect elements of AXMEDIS object

In the following those aspects will be described and solutions are proposed for them. After that, class implementation and interaction will be described.

7.2.1 AXMEDIS tool registration and certification

In the following the needed information to reach an adequate level of protection will be identified and described. For each identified information, the responsible component is found out and the relationship to the other is depicted. At the end, the interactions and operations among all involved components to guarantee trustiness of a tool are described.

7.2.1.1 Software and hardware fingerprint

Given a device and an installation of an AXMEDIS compliant software on it, a fingerprint estimation of the whole tool (software/installation and hardware/device together) is possible. In particular, this section refers to software which embed an AXOM (and thus a Protection Processor), e.g. AXMEDIS editor, compositional/formatting engine, plug-ins for external viewer/editor, etc. The certification of plug-ins for AXOM (i.e. which do not contain an AXOM) is discussed later. By fingerprint estimation is intended an extraction of relevant information regarding the device and the most important files of the application (i.e. those files which are fundamental for the trustiness of the environment). The proposed fingerprint is composed by the following data:

1. for each hard-disk in the device: serial, controller revision number
2. for each processor in the device: name (i.e. the standard description of its features), serial (if reachable)
3. BIOS: name (comprehensive of the version), serial
4. optionally, for each network device: MAC address
5. operative system: name, version, installed upgrade (e.g. SP1), serial (e.g. product id)
6. manufacturer-defined name of all available components e.g. video device, audio device, motherboard, etc...
7. For each of the following files:
 - a. executable file or library containing the AXOM
 - b. main file of each plug-ins
 - c. configuration files
 - d. secure cache filesthe following features have to be collected:
 - i. full name (path and file name)
 - ii. physical position (e.g. if mass storage is an hard-disk it is the cluster index)
 - iii. digest (e.g. MD5)
 - iv. creation date and time
 - v. last modification date and time
 - vi. size
8. AXMEDIS Tool Type ID (AXTTID)
9. AXMEDIS Registration Tool ID (AXRTID) which is the digest of the main program executable file

All this information is estimated by the Protection Processor. It is responsible of their estimation, they are stored on the AXMEDIS Certifier and Supervisor and transmitted to it by means of the PMS Client. It is to point out that last modification date and time (previous point 7-v) and size (previous point 7-vi) should not be considered as parts of the fingerprint for those file which change during the lifetime of a tool, e.g. configuration file and secure cache, otherwise information stored on the AXCS and those estimated at runtime will hardly correspond.

The fingerprint is estimated and used for the following operation:

1. Tool registration – the tool has to transmit its fingerprint to the AXCS.
2. Tool certification and authentication – the tool has to re-estimate the fingerprint and to transmit it (or a digest of it) to the AXCS

7.2.1.2 Tool certificate

Tool certificate is issued by an AXCS to the tool itself when the latter certifies itself at the first activation. The certificate is formatted in the X.509v3 format. It contains the following information:

DE3.1.2A – Framework and Tools Specification (General and Model)

Version	2 (that is X.509v3 is identified by this value)
SerialNumber	The serial number of the certificate. It is defined by the issuer which is the AXCS
Signature	The encryption algorithm used to encrypt the signature. In this case, the signature algorithm is the RSA with SHA-1 which is identified by the following ASN.1 object identifier: sha-1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
Issuer	The AXCS which issued the certificate expressed in the X.500 format
Validity	The validity period of the certificate which for the AXMEDIS purposes could be one/five year since the issuing time
Subject	Identified who or what receives the certificate. In this case it is the AXTID which identifies the tool
SubjectPublicKeyInfo	The encryption algorithm used to generate the public key and the public key itself. In this case, the used encryption algorithm is the RSA which is identified by the following ASN.1 object identifier: rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }
Encrypted	The encrypted digital signature of the certificate
Extensions	Extension to the certificate. This field can contain the enabling code for the tool and other information

Before generating the certificate, the AXCS requests to the Key Generator (component of the PMS Server) to generate a key pair (public/private keys). As said above, the AXCS inserts the public key into the tool certificate and sends certificate and private key to the tool. They are packed together as stated in PKCS #12. The integrity of transmitted information is guaranteed by a signature using the private key of the AXCS (see PKCS #12 – Public-key integrity mode). The privacy of that information is guaranteed because it is encrypted using the public key of the user who is registering the tool (see PKCS #12 – Public-key privacy mode).

The certificate and the private key are stored in the specific device certificate repository and in the PMS Client secure cache. In that way, their consistence can be tested every time the tool is used, the Protection Processor is in charge of doing that check.

The private key corresponding to the tool certificate is marked as un-extractable (see PKCS #11), i.e. it can be used on the device where it have been stored on the first time but it cannot be exported on other devices, not even by the device administrator. The tool certificate have to be accessible from all the user of a device to avoid multiple registration of the same tool by different users.

7.2.1.3 User certificate

User certificate is issued by an AXCS to the user at the registration time. It is useful to recognize who is using a tool or is making action on an object. The certificate is formatted in the X.509v3 format. It contains the following information:

Version	2 (that is X.509v3 is identified by this value)
SerialNumber	The serial number of the certificate. It is defined by the issuer which is the AXCS
Signature	The encryption algorithm used to encrypt the signature. In this case, the signature algorithm is the RSA with SHA-1 identified by the following ASN.1 object identifier: sha-1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
Issuer	The AXCS which issued the certificate expressed in the X.500 format
Validity	The validity period of the certificate which for the AXMEDIS purposes could be one/five year since the issuing time
Subject	The AXUID of the registered user
SubjectPublicKeyInfo	The encryption algorithm used to generate the public key and the public key itself. In this case, the used encryption algorithm is the RSA which is identified by the following ASN.1 object identifier: rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }
Encrypted	The encrypted digital signature of the certificate

Extensions	Extension to the certificate. This field can contain the email of the user and other information
-------------------	--

Before generating the certificate, the AXCS requests to the Key Generator (component of the PMS Server) to generate a key pair (public/private keys). As said above, the AXCS inserts the public key into the user certificate and sends certificate and private key to the user. They are packed together as stated in PKCS #12. The integrity of transmitted information is guaranteed by a signature using the private key of the AXCS (see PKCS #12 – Public-key integrity mode). The privacy of that information is guaranteed because it is encrypted using something the user knows, e.g. a password given by the user during the registration (see PKCS #12 – Password privacy mode).

The certificate can be delivered to the user using the email address he gave as reference during the registration.

Another suitable solution is to send certificate and private key (possibly packed together) to the user at the end of the registration through a secure channel, e.g. using secure protocol https (http on SSL) instead of the un-secure http.

The certificate and the private key are stored in the specific device certificate repository and in the PMS Client secure cache. In that way, their consistence can be tested every time the tool is used, the Protection Processor is in charge of doing that check.

User certificate import and export have to be somehow controlled, i.e. if a user tries to export a certificate the system have to check if he/she is the subject the certificate was issued to. For example, the PKCS #12 formatted packet the user receives at registration time is protected by the password he/she gave at that time. The main issue is to be sure that an user cannot extract a certificate installed on a device without proving he/she is really the certificate owner.

7.2.1.4 User Identifier/Identification

As stated in the previous sub-section, the AXUID (AXMEDIS User Identifier) of a user is contained in his/her certificate. On a multi-user device (e.g. a personal computer with MS Windows), several certificates for different users can be stored. In that scenario, the main issue is how an AXMEDIS tool can get the right certificate (thus the right AXUID) for the current user.

Usually a multi-user device manages one certificate repository for each system users. In that case, the AXMEDIS tool can look for a certificate issued by the AXCS in the current user certificate repository and use that certificate, if exists, as reference certificate for the current system user. Note that each user should have only one personal certificate issued by the AXCS.

The Protection Processor is in charge of looking for the certificate in the system repository and to get the contained AXUID which univocally identified the AXMEDIS user.

7.2.1.5 Date and time

Since licenses can be based on date/time condition, date and time control is a fundamental issue to be addressed.

Date and time have to be measured at the registration and every time an action is made. These measures (or at least the last one) should be stored in a trusted place which can be suitably the AXCS.

In that way, during certification and authentication, the AXCS can verify if the nominal sequence of the actions (i.e. the sequence according to which actions were stored) matches the timeline sequence (obtainable using the measured date and time).

Each record of the history of actions (see below) have to be labelled with the date and time of execution.

7.2.1.6 Action history

Action history permits to control if the user made some not-allowed actions during off-line working. That is, when the tool is used on-line (i.e. it can freely communicate with the PMS Server) every time the user request to do an action on an object the request is processed and, if it is authorized, an action log (containing the action type, the user id, the tool id, date and time, etc...) is sent to the AXCS. On the other hand, if for a while the tool has been used off-line (i.e. without communication to and from the PMS Server), each authorized action generates an action log which is cached on the device. As soon as a connection to the PMS Server is available, the set of cached action logs is sent to it.

The information involved in this kind of control is:

1. the action logs which are stored on the device

DE3.1.2A – Framework and Tools Specification (General and Model)

2. a hash (e.g. MD5) of the past history of actions which is stored on the AXCS and on the device. Every time the tool can communicate with the PMS Server, the former sends to the latter the cached action logs (or a single action log if it is working on-line) and the updated hash of the history, i.e. a hash which is function of the old hash and the cached action logs.

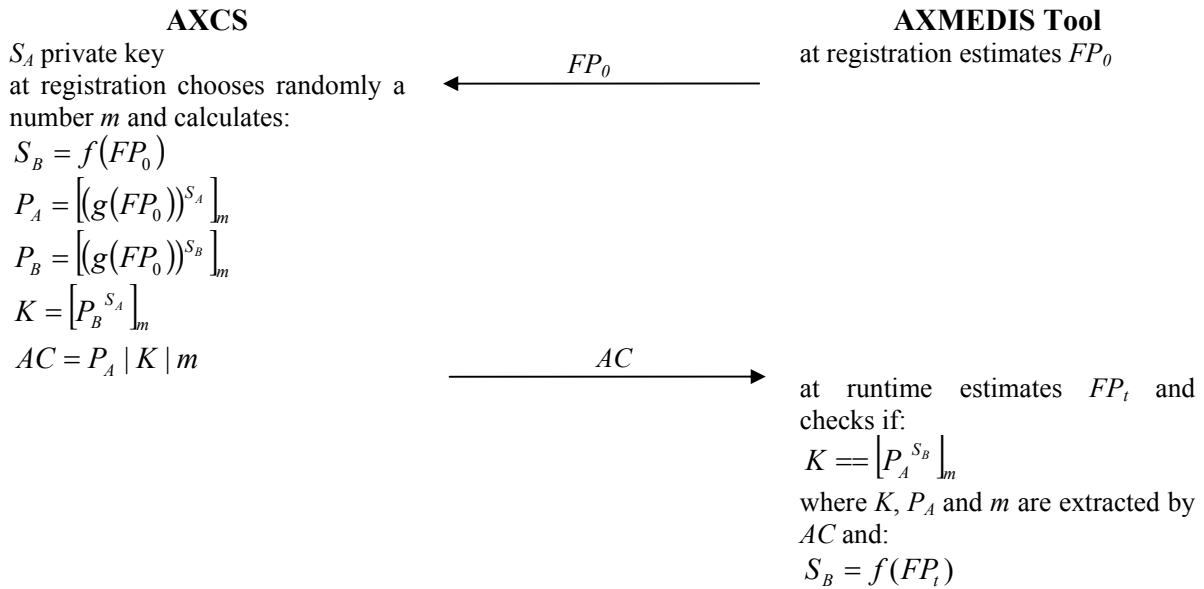
7.2.1.7 Enabling code

Enabling code is issued by the AXCS to the tool during the registration (it could be stored in the tool certificate). The code is function of the fingerprint of the device. The generation function is secret but the tool is capable to test if its actual fingerprint matches the one at registration time. In that way, the tool can control itself if something is changed (hardware or software) since the registration.

The enabling code mechanism exploits the theory of Diffie-Hellman on keys generation. In the following figure the code generation function and test function are depicted. At registration time, the tool estimates its fingerprint (FP_0) and sends it to the AXCS. It calculates the enabling code (AC) for that fingerprint as depicted in the picture and sends it back. At runtime, the tool re-estimate its fingerprint and verifies if it is compatible with the stored AC using the test explained in the picture. Notice that the enabling code is generated using S_A and the function $g(FP)$ which are not used (and available) by the tool to make the test. The common information between the tool and the AXCS is AC and the function $f(FP)$ therefore the tool is not able to re-calculate a new AC if its fingerprint changes.

The activation number is a way to verifies tool integrity during off-line working.

To make the enabling code calculation and test easier instead of using the entire fingerprint it is better to use a hash/digest of it, e.g. and MD5.



In the formulas in the figure, square brackets subscribed with m is the base m modulus operation.

S_A, g, f, m have to be someway chosen, e.g. they can be randomly created for each tool registration and stored in the related record.

7.2.1.8 Trustiness of a tool

In this sub-section, a feasible mechanism to test the trustiness of a tool (the pair application/device) is depicted. The information described above is involved in that check.

		Protection Processor	PMS Client	PMS Server	AXCS
Fingerprint	registration	estimates FP_0 ;	-	-	stores FP_0 ;
	time t	estimates FP_t ;	joins FP_t to action logs;		verifies FP_t w.r.t. FP_0 ;
Tool certificate		verifies	stores; joins TID to action logs;	generates keys:	generates certificate:

DE3.1.2A – Framework and Tools Specification (General and Model)

User certificate	verifies	stores;	generates keys;	Generates certificate;
User Identifier	gets	joins to action logs;	-	-
Date and time	measures;	joins to action logs;	-	verifies; stores date and time of the last action log;
Action history	estimate new history hash;	manages action logs; stores history hash;	-	verifies action logs; stores new history hash;
Enabling code	tests;	stores;	-	generates as function of FP_0 .

In the above table, all task of each involved component are reported with respect to the needed information, i.e. which component generates/estimates an information and which other uses/verifies it.

If a tool have been successful registered by an user, at runtime the tool is considered trusted if, and only if, all the following tests succeed:

1. The user certificate is valid, i.e. it is digitally signed by the certification authority (which is an AXCS). Control performed by the Protection Processor;
2. The tool certificate is valid. Control performed by the Protection Processor;
3. The enabling code contained in the tool certificate and the re-estimated fingerprint are compatible according to the test described above. Control performed by the Protection Processor;
4. The re-estimated fingerprint digest matches the registration-time fingerprint digest which is stored on the AXCS. Control performed by the AXCS;
5. The history hash calculated by the Protection Processor, using the old one and the action logs (both securely stored by the PMS Client), coincides with the same hash estimated by the AXCS using its own copy of the old history hash and the action logs received;
6. The execution date and time of all the action listed in the action logs are consistent each other (i.e. the nominal order matches the timeline order) and with respect to the last action execution date and time stored on the AXCS;

7.2.1.9 Certified software

AXMEDIS tool features can be enriched by means of plug-ins. Usually plug-ins are pieces of software exposing functions for specific purposes. Thus it is not suitable to equip a (likely) simple piece of software (as a plug-in can be) with the AXOM to ensure DRM respect. Nevertheless, a plug-in can be used to manipulate DRM-labile data, e.g. a fingerprint extractor plug-in.

To tackle this lack of security, each pieces of software created to enrich AXOM functionalities via plug-in and to manage content have to be previously certified by the AXCS. AXMEDIS certification consists of the following steps:

1. The final version of a software in binary form (e.g. a DLL) is submitted to the AXMEDIS certifiers (human beings) declaring what are the purposes of that software;
2. The certifiers analyze the functions of the software and verify that it does not make unfeasible actions, e.g. backup the unprotected content out of trusted environment;
3. If the software conforms to AXMEDIS guidelines and respects what have been initially declared it is approved;
4. AXCS creates a certificate for the approved software. The information contained in it is described below;
5. The approved software to be recognize as AXMEDIS compliant by an AXOM have to expose the given certificate.

The certificate contains the information summarized in the table below. It is formatted as SignedData type of PKCS #7.

Digest and encryption algorithm	The signature algorithm is the RSA with SHA-1 identified by the following ASN.1 object identifier: sha-1WithRSAEncryption OBJECT IDENTIFIER ::=
--	--

DE3.1.2A – Framework and Tools Specification (General and Model)

	{iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
Issuer and certificate serial number	The distinguished name which refers to the AXCS (X.500 format) and the corresponding certificate serial number
Encrypted digest	The encrypted digital signature of the software this certificate refers to

During software installation, the related certificate have to be stored in a repository reachable by all AXMEDIS tool installed on that device, e.g. a list of entries in the system registry. In that way, plug-ins are shared among all tools which can use them.

Every time a piece of protected content have to be passed to an external software, the Protection Processor controls if the binary file containing the software corresponds to the signature contained in the declared certificate during installation.

7.2.1.10 Execution controls

Protection Processor should inhibit AXMEDIS tool functioning as soon as it reveals that tool execution is under tracking. That is, Protection Processor should test if execution is under debug. That control is useful to avoid malicious user to disclose all security mechanism used within AXOM to guarantee trustiness of the environment.

This feature is system-dependant and it will be specifically developed for each platform AXMEDIS Tool should be capable to run on.

7.2.2 Robustness against malicious user actions

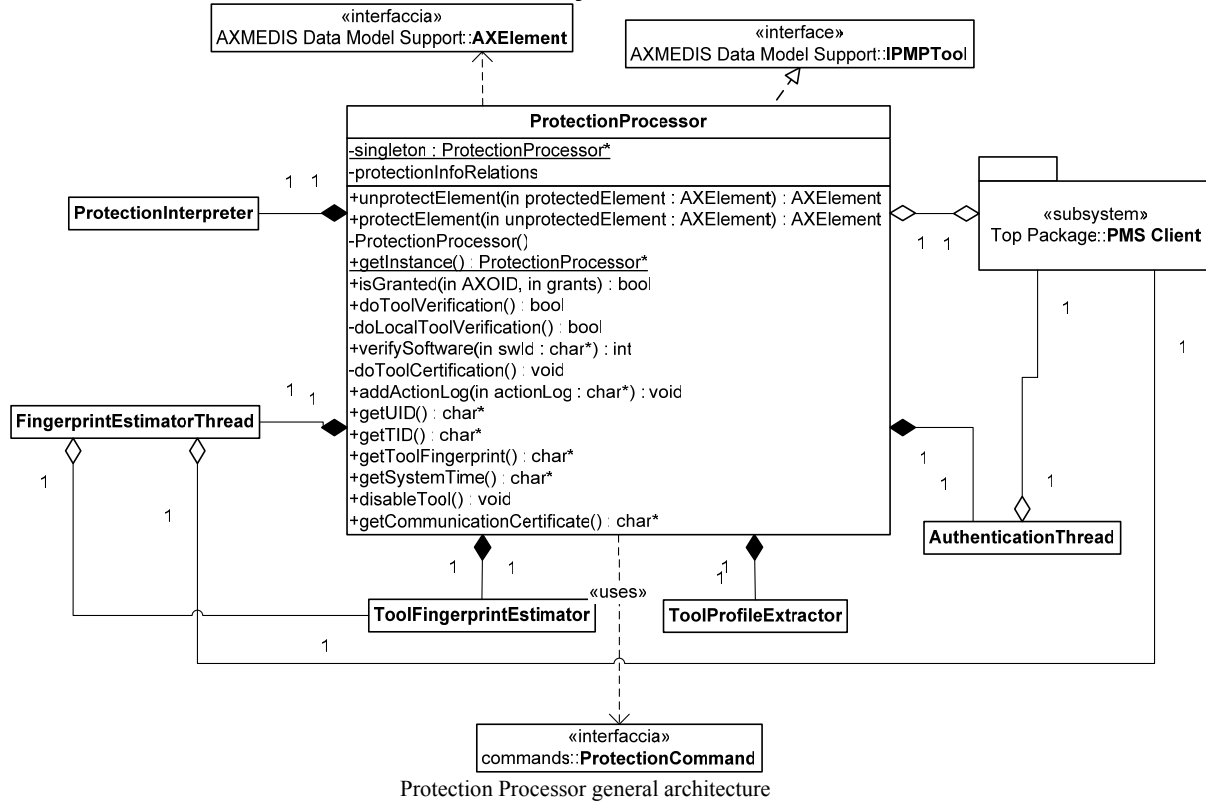
In the following table are listed several feasible actions which a malicious user can carry out to illegally use DRM-controlled content. Each attack is shortly described and

Feasible attacks description	System avoid/detect capabilities
Migrate an AXMEDIS tool installation from the registered device to another	The software and hardware fingerprints of the two devices are not equal, even if the two device has been assembled with same components, thus enabling code will fail.
Software under debug during execution	Protection processor monitors if the application embedding it is under debug, if it is protection processor will clear all sensible data on the device. Moreover, if this monitoring system is deactivated modifying the application file, the fingerprint of the tool will change and it will stop working.
Change system time to use time-constrained content out of DRM-allowed period	This attack will work until the device works off-line. As soon as the tool can communicate with an AXCS thus, if IPR infringement is detected by the latter, the tool will be disabled and the sensible information will be cleared.
Migration of tool certificate	There are two countermeasures against this action: <ul style="list-style-type: none"> • The certificate is stored in two different places: the system certificate repository and the secure cache of PMS Client. The user can move the one stored in the system repository to another device but a simple consistency control between the two copies will defeat the attack • The certificate contains the signed activation code used to verify the tool integrity. Therefore, a certificate copied by another device will never work
An user uses content by illegally exposing the identity of another user	If a user exposes all the credential of another user the attack will be not detect. Nevertheless, this attack is similar to the usage of stolen credit card and it cannot be considered solvable problem: the privateness of the personal certificate is exclusive interest of the user himself/herself
Change pieces of hardware	Depending on the wideness of changes, the tool is disabled and a new registration is requested
Create a backup of the a newly-installed AXMEDIS tool. Illegally use content off-line (i.e. without	If a ghost image of the hard disk is made before using the content and then it is restored on the original hard-drive after content has been illegally used off-line, the attack will not be revealed by the

external controls). Restore the virgin version of the tool before re-connect to the network.

protection processor or the AXCS. Nevertheless, this action require such technical support and knowledge that it is not applicable by the common user and it do not allow a spread distribution of unprotected content. Thus it is not of interest for us.

7.2.3 Protection Processor Class hierarchy



ProtectionProcessor is the main class of the Protection Processor package. It provides protection services to the Data Model Support, the Command Manager and the PMS Client. As described in the previous subsections, it is in charge of:

1. certifying and verifying an AXMEDIS tool
2. controlling software which uses sensible content and does not contain AXOM
3. revealing attacks during tool execution
4. protecting and un-protecting AXMEDIS elements

As depicted in the picture above, **ProtectionProcessor** is the core of this package. It is the connection point among all other utility (protection information interpretation, etc...). For each of the above-listed macro functionalities the related functions and classes are shortly described below.

Certification and verification

doToolCertification – this function is called at first tool activation to certify it, sending the required information to the AXCS.

doToolVerification – verifies tool integrity by calling *doLocalToolVerification* and sending verification information to the AXCS.

doLocalToolVerification – controls: that tool fingerprint and the enabling code match, integrity of the used certificates, etc...

DE3.1.2A – Framework and Tools Specification (General and Model)

addActionLog – this function has to be called by the PMS client every time a grant is requested. In that way protection processor can estimate a checksum for the overall history of the performed actions with a given tool.

getToolFingerprint – returns the fingerprint of the tool.

ToolFingerprintEstimator – this class exposes a method which estimates the fingerprint of the tool (hardware and software) and returns it as XML string. Obviously, this class is system-dependant because it has to use OS/HW-dependant functions to retrieve the required information to estimate the tool fingerprint.

ToolProfileExtractor – this class exposes a method which extracts the profile of the tool (hardware and software) and returns it as XML string. Obviously, this class is system-dependant because it has to use OS/HW-dependant functions to retrieve the required information to extract the tool profile.

Software verification

verifySoftware – given a software identifier (e.g. a path or something else), it verifies that the software has a valid certificate and that the software hash correspond to that certificated by the authority.

Tool execution control

AuthenticationThread – this is a thread which randomly controls tool integrity. That is, it controls the application is not under debug, it calls *doLocalVerification* and, if possible, it sends some control information to the AXCS.

Protection and un-protection of AXMEDIS element

unprotectElement – given a protected AXMEDIS element, this function retrieves the related protection information and unprotect the protected content obtaining the corresponding clear-text AXMEDIS element.

protectElement – given an unprotected AXMEDIS element and the list of instructions to protect it, this method protects the element obtaining a protected AXMEDIS element and the related protection information.

ProtectionInterpreter – this classes is capable to parse protection information to instantiate a set of **ProtectionCommand** which can be used to unprotect a given element.

ProtectionCommand – this classes is the super-class for all those class which represent protection commands, it exposes the basic methods needed to protect and unprotect content.

Other functions and utilities

These are general functions exposed by **ProtectionProcessor** which are used in several parts of the security system.

getInstance – the design pattern singleton is applied to that class

getUID – since **ProtectionProcessor** manage the identity of the user who uses the tool, this function allows the other components to retrieve the unique identifier of that user. In particular, this function should be used by the PMS Client to determine the AXUID to be added to an Action Log.

getTID – since **ProtectionProcessor** is in charge of certifying the tool, this function allows the other component to obtain the unique identifier of the tool itself. In particular, this function should be used by the PMS Client to determine the AXTID to be added to an Action Log.

getSystemTime – this function allows the tool components to obtain the controlled system time in a unique manner. In particular, this function should be used by the PMS Client to determine the execution time of an action to be added to an Action Log.

DE3.1.2A – Framework and Tools Specification (General and Model)

getCommunicationCertificate – since **ProtectionProcessor** manages all the certificates involved in the security system of the AXMEDIS tool, this function is the unique way to obtain a communication certificate valid to exchange data with the server-side components of the AXMEDIS framework, e.g. PMS Server or AXCS.

disableTool – does all the needed action to disable a tool. That is, it removes all assigned certificates and the enabling code, it clear the cache of the protection information of the PMS Client, all the sensible information, etc... This function should be called every time a lack of integrity is detected.

FingerprintEstimatorThread – this thread is needed because tool fingerprint estimation can be burdensome for the device and it cannot be executed every time is required (e.g. at each user action). In that way, the fingerprint is randomly estimated at runtime without stopping program execution.

7.2.4 Protection Info and Procedure Interpreter (DSI)

Module Profile		
Protection Info and Procedure Interpreter		
Executable or Library(Support)	Static library	
Single Thread or Multithread	Single Thread	
Language of Development	C++	
Responsible Name	Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
MPEG-21 Part 4: Intellectual Property Management and Protection Components		w6772 http://mpeg.nist.gov/
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

This tool is capable of processing the IPMP information of MPEG21 according to the standard and much more. The information may include:

- How each element of an AXMEDIS object has been protected, i.e. encrypted, encoded, compressed and scrambled
- How each chunk of a resource has been protected, e.g. specifying that a given set of protection tools has to be applied from byte X to byte Y of a given resource (and not to the whole resource). In that way, different protection can be applied to a resource along its consumption.
- It is based on an XML schema which allows to describe sort of protection procedures

For instance, suppose the resource is an image comprised of two blocks, it can be protected as follow:

Block 1, from 0 to 1234 byte

- Unzip(.....parameter)
- Decrypt(.....parameters)
- Descramble (.....parameters....)

Block 2, from 1235 to the end of the file

- Descramble (.....parameters....)
- Unzip(.....parameter)
- Decrypt(.....parameters)
- Unzip(.....parameter)

DE3.1.2A – Framework and Tools Specification (General and Model)

- Descramble (.....parameters.....)

Different tools can be selected by means unique identifiers defined in the framework. The tools can on the device, downloaded from some server or directly contained within the protection info.

The decomposition and the application of the dynamic IPMP can be performed at level of data segments and blocks and may change over time if different coding protection models are enforced into the same resource. This is quite different with respect to dynamically change the IPMP rules when the resource is streamed.

Note that the single resource may have different tools, different keys, different combination or tools, etc...

Protection command used by the Protection Interpreter are distributed as dynamic library (e.g. DLL for windows, SO for Linux, etc...). Each dynamic library which contains a protection command exposes a given interface (see below) and its content (a protection command) is described by description file (XML file as stated for plug-ins).

7.2.4.1 Protection Info Format (DSI)

Protection information are formatted as stated in MPEG-21 Part 4 IPMP standard. The syntax and semantics is still under discussion, the actual state of the standard is contained in the output document w6772 of the 70th MPEG meeting (see <http://mpeg.nist.gov/>).

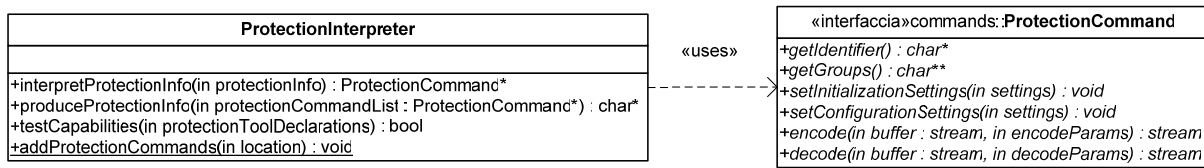
MPEG-21 Part 4 divides protection information into two XML schemas:

- one is used to declare the list of needed protection tools (or commands as defined in this section) to unprotect the whole digital item;
- the other is used to describe, for each protected element, how to use those tools (e.g. the execution order, keys, initialization parameters, etc...) to unprotect a specific element.

The former part of protection information (i.e. the list of all needed tools) should not only contain the necessary tools to unprotect the “first level” of protected element, it should contain also the required tools to correctly manage all nested levels of protected elements. In that way, looking at the tool list declaration it will be possible to immediately decide whether an AXMEDIS Tool is capable to completely “consume” an object.

7.2.4.2 Protection Interpreter (DSI)

Protection Interpreter is the part of Protection Processor in charge of transform an XML description of the protection information into instance of **ProtectionCommand** and vice versa.



Protection Interpreter main classes

As depicted above, this package contains two main classes:

- **ProtectionInterpreter** – exposes functionality to interpret protection information and to verify capabilities for un-protecting an AXMEDIS object. It also exposes methods for the general management of protection tools installed on an AXMEDIS tool.
- **ProtectionCommand** – is a common interface for all those classes which represent protection tools (in the MPEG-21 terminology). Fundamentally, it exposes function to decode/encode a bundle of bytes (e.g. represent by a stream) and all needed functions to manage information contained within the protection information.

In the picture below, the interaction among Protection Processor, Interpreter and PMS Client is sketched. As depicted, **ProtectionInterpreter** does not directly unprotect an element, it only creates a sequence of commands which will be used by the **ProtectionProcessor** on a specific element obtaining a clear text element.

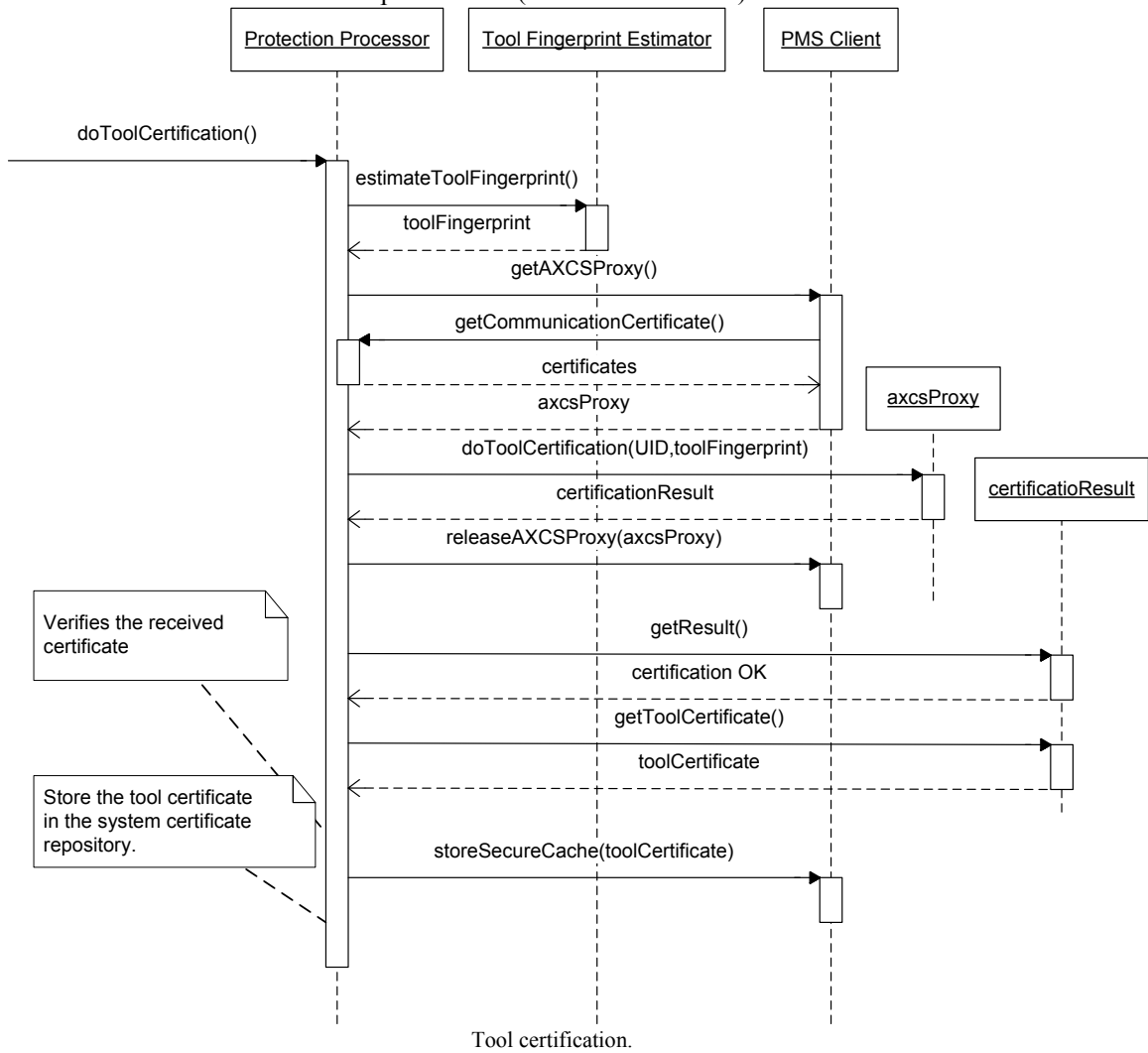


7.2.5 Tool certification/registration (DSI, FUPF)

Module Profile		
Tool Certification		
Executable or Library(Support)	Static library	
Single Thread or Multithread	Single Thread	
Language of Development	C++	
Responsible Name	Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces-C++	Version 2.6.0	Apache License, Version 2.0

During tool certification, the protection processor estimates the tool fingerprint which is comprehensive of device fingerprint, tool installation fingerprint and AXRTID. After that, it requires a communication channel towards the AXCS to the PMS client. Through that channel, protection processor sends to AXCS the fingerprint and the AXMEDIS User ID of the user who is registering the tool. AXCS returns a CertificationResult (see Part H “Protection and Accounting Tools” of specification) which contains a status variable, whose value depends due to the server side verification of the sent data, and, if certification succeeds, a certificate for the tool containing the TID and the enabling code.

DE3.1.2A – Framework and Tools Specification (General and Model)



7.2.6 Tool verification/authentication (DSI)

Module Profile		
Tool Registration		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Responsible Name	Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux, MACOS X	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces-C++	Version 2.6.0.	Apache License, Version 2.0

The figures below depict the interaction among command manager, protection processor and PMS client during content consumption and handling in the following cases:

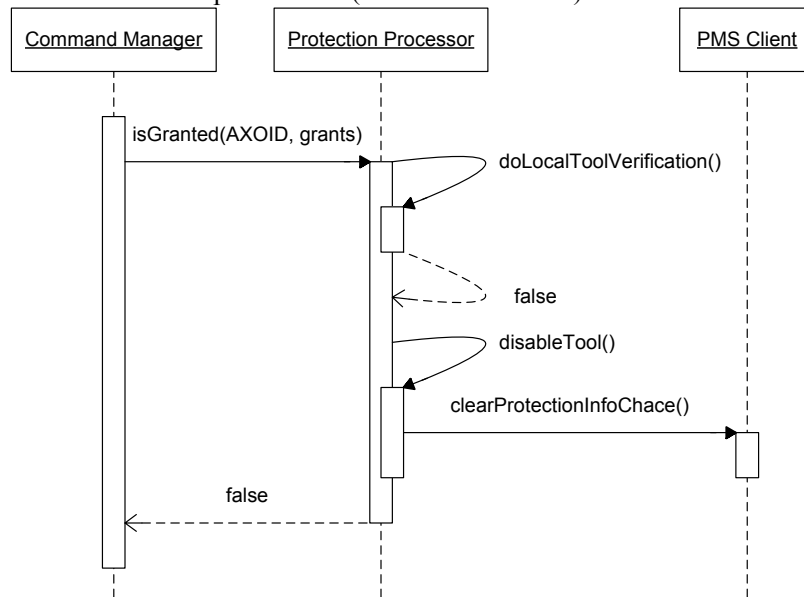
- Local tool verification fails
- Remote tool verification fails
- Verifications succeed and the garnt request is delegated to the PMS Client

As stated above and depicted in the figures, protection processor uses a proxy to communicate the verification information to the AXCS. That data consist of:

- TID – the tool identifier assigned to the tool at certification time by the AXCS
- Tool fingerprint hash – to make the commuication lighter, the whole tool fingerprint is transmitted only if its hash do not match the one stored on the AXCS
- Time – the timestamp of the verification request
- New history hash – the new hash of the action log history estimated on the client. It ha sto be re-estimated by the server
- Action log list – a list of all action log stored during off-line use of the client

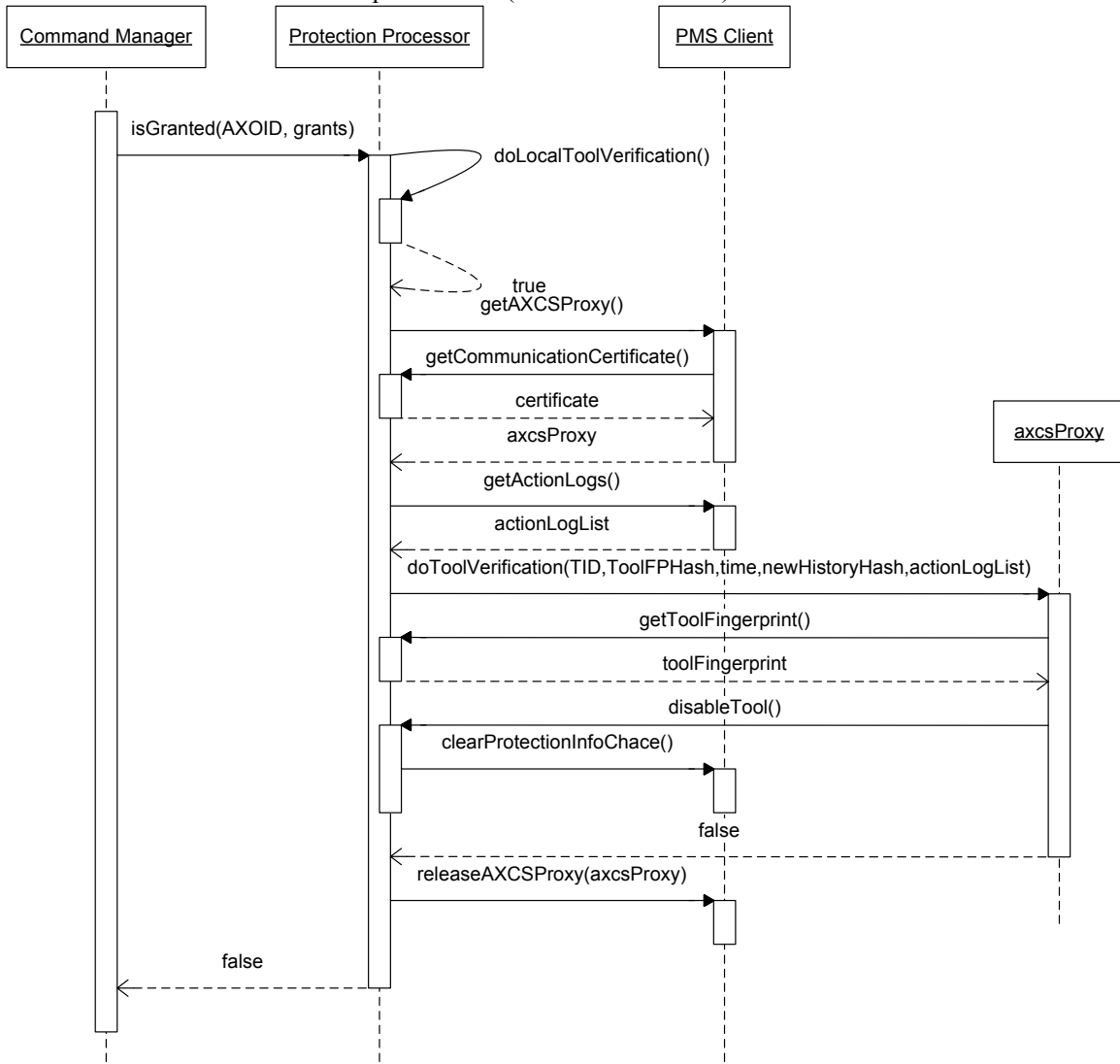
Communication between protection processor and AXCS works on a secure channel provided by the PMS Client. It has to be pointed out that even if communication is established by the PMS client certificates for communication are managed by the protection processor (as all the other security information) thus PMS client has to request the communication certificates to the protection processor every time it has to open a secure channel.

DE3.1.2A – Framework and Tools Specification (General and Model)



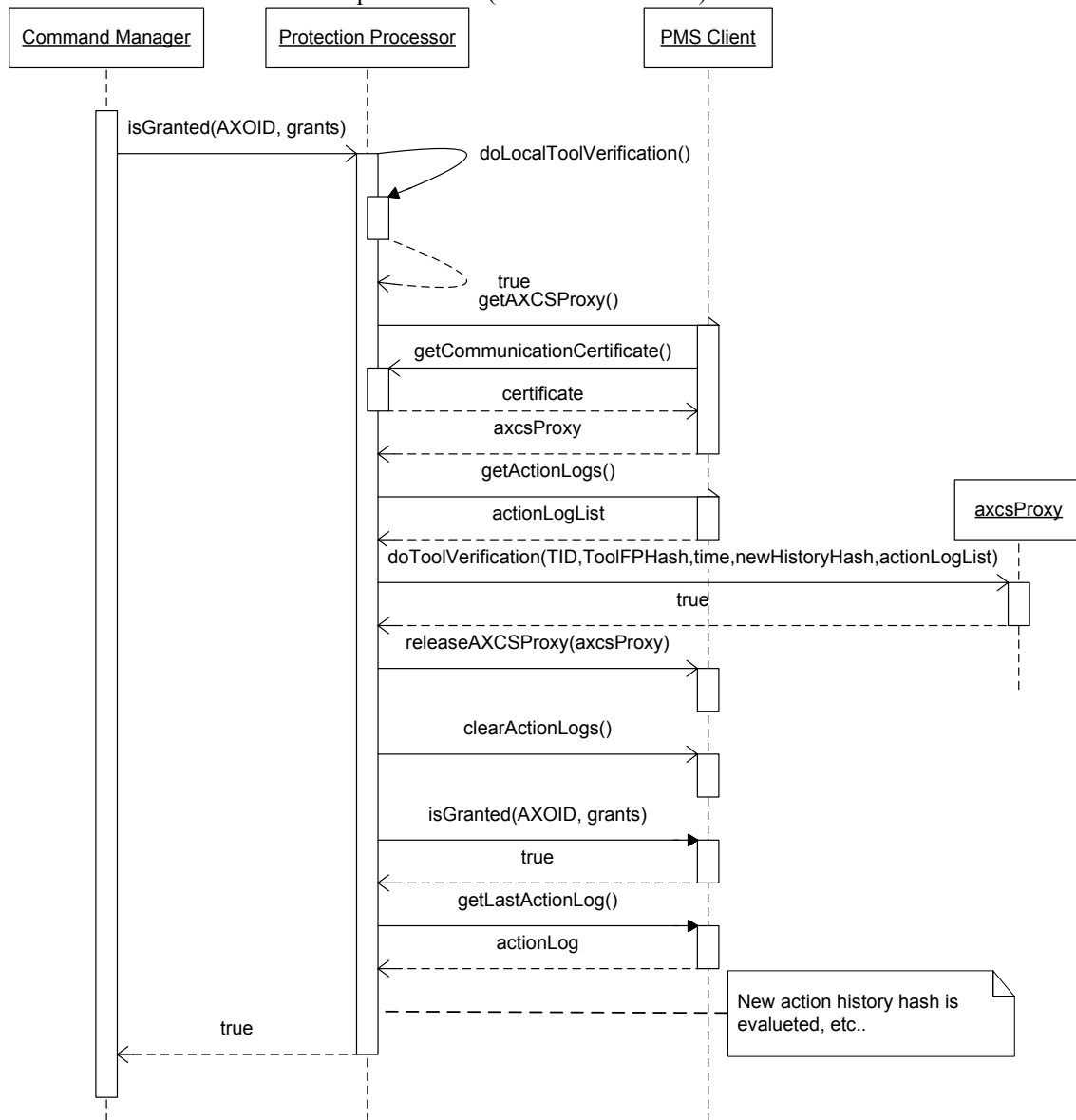
Interactions among Command Manager, PMS Client and Protection Processor during grant authorization request which fails because tool does not pass local verification (e.g. enabling code).

DE3.1.2A – Framework and Tools Specification (General and Model)



Interactions among Command Manager, PMS Client and Protection Processor during grant authorization request which fails because tool does not pass server-side verification.

DE3.1.2A – Framework and Tools Specification (General and Model)



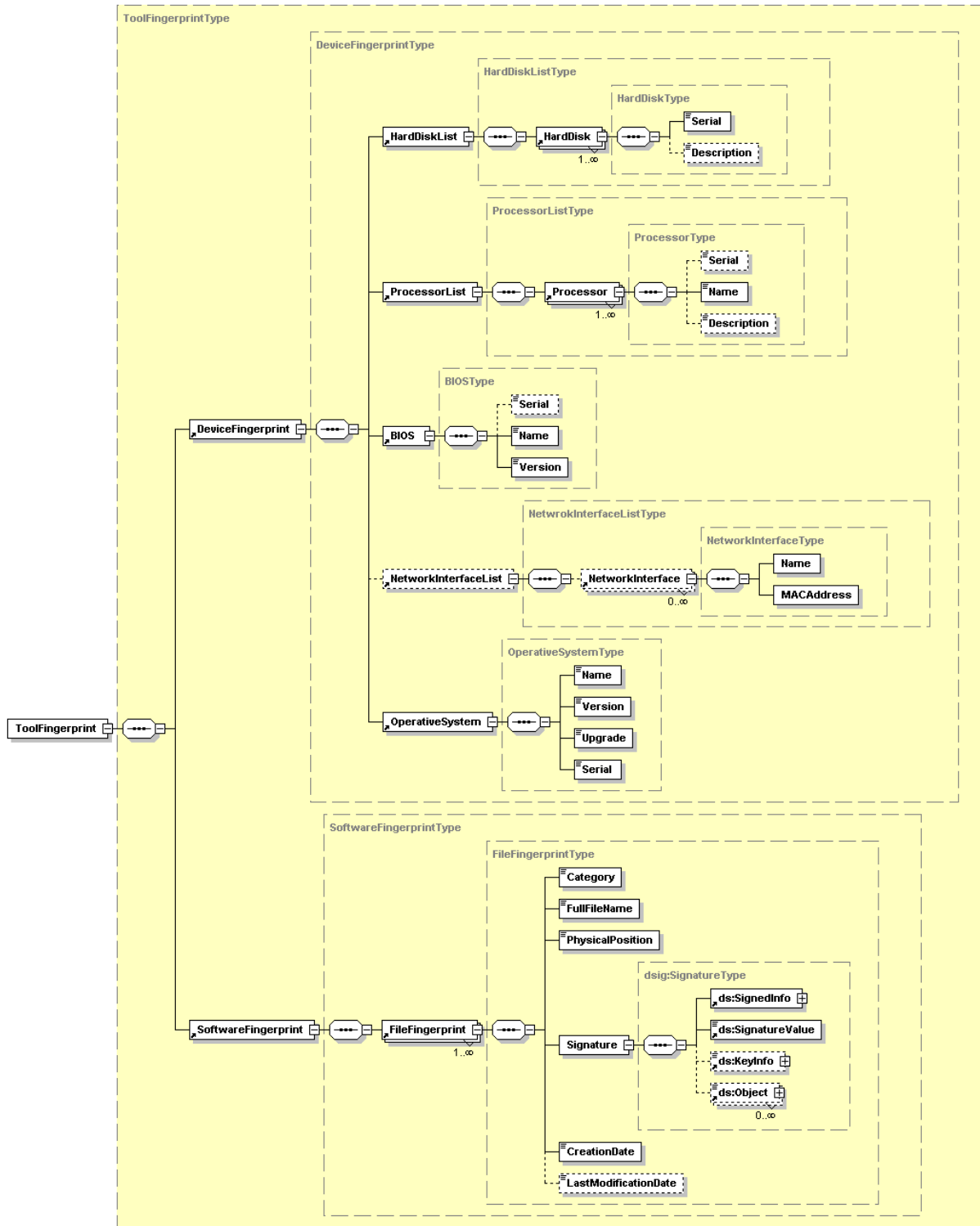
Interactions among Command Manager, PMS Client and Protection Processor during grant authorization request which succeeds.

7.2.7 Tool Fingerprint Estimation on PC (DSI)

Module Profile		
Tool ID Estimation		
Executable or Library(Support)		
Single Thread or Multithread		
Language of Development		
Responsible Name		
Responsible Partner		
Status (proposed/approved)		
Platforms supported		
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

Tool fingerprint is described in the sections above, it contains information on the hardware (i.e. the personal computer) and on the software. It has been created to uniquely identify an installation of a given AXMEDIS-compliant application on a given device and to allow detection of software/hardware changes. Fingerprint is stored and transmitted as XML file/message with the following schema:

DE3.1.2A – Framework and Tools Specification (General and Model)



Toolfingerprint XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.axmedis.org/tool-fp" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.axmedis.org/tool-fp" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="xmldsig-core-schema.xsd"/>
  <xs:element name="ToolFingerprint" type="ToolFingerprintType"/>
  <xs:complexType name="ToolFingerprintType">
    <xs:sequence>
```

```

        <xs:element ref="DeviceFingerprint"/>
        <xs:element ref="SoftwareFingerprint"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="DeviceFingerprint" type="DeviceFingerprintType"/>
<xs:complexType name="DeviceFingerprintType">
    <xs:sequence>
        <xs:element ref="HardDiskList"/>
        <xs:element ref="ProcessorList"/>
        <xs:element ref="BIOS"/>
        <xs:element ref="NetworkInterfaceList" minOccurs="0"/>
        <xs:element ref="OperativeSystem"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="HardDiskList" type="HardDiskListType"/>
<xs:complexType name="HardDiskListType">
    <xs:sequence>
        <xs:element ref="HardDisk" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="HardDisk" type="HardDiskType"/>
<xs:complexType name="HardDiskType">
    <xs:sequence>
        <xs:element name="Serial" type="xs:string"/>
        <xs:element name="Description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="ProcessorList" type="ProcessorListType"/>
<xs:complexType name="ProcessorListType">
    <xs:sequence>
        <xs:element ref="Processor" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Processor">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ProcessorType"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:complexType name="ProcessorType">
    <xs:sequence>
        <xs:element name="Serial" type="xs:string" minOccurs="0"/>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="BIOS" type="BIOSType"/>
<xs:complexType name="BIOSType">
    <xs:sequence>
        <xs:element name="Serial" type="xs:string" minOccurs="0"/>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Version" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="NetworkInterfaceList" type="NetworkInterfaceListType"/>
<xs:complexType name="NetworkInterfaceListType">
    <xs:sequence>
        <xs:element ref="NetworkInterface" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="NetworkInterface" type="NetworkInterfaceType"/>
<xs:complexType name="NetworkInterfaceType">
    <xs:sequence>
        <xs:element name="Name"/>
        <xs:element name="MACAddress"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="OperativeSystem" type="OperativeSystemType"/>
<xs:complexType name="OperativeSystemType">
    <xs:sequence>

```

```

<xs:element name="Name" type="xs:string"/>
<xs:element name="Version" type="xs:string"/>
<xs:element name="Upgrade" type="xs:string"/>
<xs:element name="Serial" type="xs:string"/>
</xs:sequence>
</xs:complexType>
<xs:element name="SoftwareFingerprint" type="SoftwareFingerprintType"/>
<xs:complexType name="SoftwareFingerprintType">
  <xs:sequence>
    <xs:element ref="FileFingerprint" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="FileFingerprint" type="FileFingerprintType"/>
<xs:complexType name="FileFingerprintType">
  <xs:sequence>
    <xs:element name="Category">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="CONTAIN_AXOM"/>
          <xs:enumeration value="PLUG_IN"/>
          <xs:enumeration value="CONFIGURATION"/>
          <xs:enumeration value="SECURE_CACHE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="FullFileName" type="xs:anyURI"/>
    <xs:element name="PhysicalPosition" type="xs:string"/>
    <xs:element name="Signature" type="dsig:SignatureType"/>
    <xs:element name="CreationDate" type="xs:dateTime"/>
    <xs:element name="LastModificationDate" type="xs:dateTime" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

As stated before, the information contained in the fingerprint are mainly information for the identification of the tool, however profile-related information can be estimated in a similar way and merged with those above reported. Samples of profile information are the following:

- Which kind of content it may manage or not, e.g. it cannot load PDF, it can load PS
- Resolution of the screen device
- Power of the device
- Presence of some basic adaptation tools or their absence
- Print capabilities or not
- Audio capabilities or not
- Video streaming capabilities or not
- Burning ROM capabilities or not
- Network connection speed
- Network connection type, e.g. permanent or irregular
- etc....

7.2.8 Tool ID Estimation on PDA (EPFL)

Module Profile		
Tool ID Estimation		
Executable or Library(Support)	Library	
Single Thread or Multithread		
Language of Development	C/C++	
Responsible Name	Zoia	
Responsible Partner	EPFL	
Status (proposed/approved)	proposed	
Platforms supported	Pocket PC 2003	
Interfaces with other tools:	Name of the communicating tools	Communication model and format

DE3.1.2A – Framework and Tools Specification (General and Model)

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

PDA devices might have built-in serial numbers that can be read programmatically. If the PDA runs Pocket PC 2002 or later the serial number can be retrieved by using the functions: SystemParametersInfo, and KernelIoControl. This sample code (from http://www.pocketpcdn.com/articles/serial_number2002.html) returns the serial number of the device:

```
#include <WINIOCTL.H>
```

```
extern "C" __declspec(dllimport)
```

```
BOOL KernelIoControl(
    DWORD dwIoControlCode, LPVOID lpInBuf, DWORD nInBufSize,
    LPVOID lpOutBuf, DWORD nOutBufSize, LPDWORD lpBytesReturned
);
```

```
#define IOCTL_HAL_GET_DEVICEID CTL_CODE(FILE_DEVICE_HAL, 21, METHOD_BUFFERED,
FILE_ANY_ACCESS)
```

```
CString GetSerialNumberFromKernelIoControl()
{
```

```
    DWORD dwOutBytes;
    const int nBuffSize = 4096;
    byte arrOutBuff[nBuffSize];
```

```
    BOOL bRes = ::KernelIoControl(IOCTL_HAL_GET_DEVICEID,
        0, 0, arrOutBuff, nBuffSize, &dwOutBytes);
```

```
    if (bRes) {
        CString strDeviceInfo;
        for (unsigned int i = 0; i < dwOutBytes; i++) {
            CString strNextChar;
            strNextChar.Format(TEXT("%02X"), arrOutBuff[i]);
            strDeviceInfo += strNextChar;
        }
        CString strDeviceId =
            strDeviceInfo.Mid(40,2) +
            strDeviceInfo.Mid(45,9) +
            strDeviceInfo.Mid(70,6);
    }
```

```

        return strDeviceId;
    } else {
        return _T("");
    }
}

```

7.2.9 Scramble/Descramble Support (EPFL)

The word “scrambling” is often mis-used as a synonym of cryptography.

The term cryptography refers to the tools and mechanisms enabling:

- Tamper detection allows the information receiver to verify that it has not been modified during transmission. If there were any attempt to modify or substitute data, a false message would be detected.
- Authentication allows the information receiver to determine who sent the message.
- Privacy/confidentiality ensures that no one can read the message except the intended receiver. Integrity assures the receiver that the message that they received was not modified in any way since it was sent from the origin.
- Non-repudiation is a mechanism that proves that the sender really sent the message.

Lastly, scrambling allows two communication parties to disguise information they send to each other. The sender encrypts/scramble the information before sending it. The receiver decrypts/descramble the information after receiving it.

In cryptographic terminology, the message is called plaintext or cleartext. Encryption is encoding the contents of the message in such a way that hides its contents from outsiders. The encrypted message is called the ciphertext. The process of retrieving the plaintext from the ciphertext is called decryption. Encryption and decryption usually make use of a key, and the coding method is such that decryption can be performed only by knowing the proper key.

Scrambling / Descrambling algorithms are based on secret key algorithms.

In secret key cryptography, a single key is used for both encryption and decryption. The sender uses the key to encrypt the plaintext and then sends the ciphertext to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. There are several widely used secret key cryptography schemes [Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR) modes] and they are generally categorized as being either block ciphers or stream ciphers. A block cipher is so-called because it encrypts blocks of data at a time. The same plaintext block will always be encrypted into the same ciphertext when using the same key. Stream ciphers operate on a single bit, byte, or word at a time, and implements a feedback mechanism so that the same plaintext will yield different ciphertext every time it is encrypted.

Usually, scrambling / descrambling algorithms refer to stream ciphers algorithms.

In the past, the scrambling process did not change the information with the content but it only “mix” it. In this way the “preview” of the content could be done with low cost device.

Scrambling algorithms used to allow “content preview” functionality are strongly linked to proprietary and “unknown” solutions and implementation. This is what in cryptography is called “security by obscurity” that it is not an “open” approach.

Due to the evolution of the attacks and the openness of the approach, AXMEDIS should use more sophisticated scrambling algorithms that use strong known secret key algorithms. A well-know library called cryptlib provides many functions, including all what may be necessary in the project.

Module Profile	
Scramble/Dscramble	
Executable or Library(Support)	Support library
Single Thread or Multithread	
Language of Development	C/C++

DE3.1.2A – Framework and Tools Specification (General and Model)

Responsible Name		
Responsible Partner		
Status (proposed/approved)	Proposed	
Platforms supported	cryptlib is supplied as source code for Unix (static and shared libraries), DOS, Windows 3.x, Windows 95/98/ME, Windows NT/2000/XP, OS/2, BeOS, Macintosh, and the Tandem environment, and also as 16- and 32-bit Windows DLL's. cryptlib is also available as an ActiveX control for Windows, and adaptations exist for VM/CMS and MVS mainframe environments.	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
cryptlib	Cryptlib 3.1	cryptlib is distributed under a dual license that allows free, open-source use under a GPL-like license and closed-source use under a standard commercial license. In addition, cryptlib is free for use in low-cost, non-open-source applications such as shareware, and for personal and research use.

Application Programming Interface

The API of cryptlib serves as an interface to a range of plug-in encryption modules that allow encryption algorithms to be added in a fairly transparent manner, so that adding a new algorithm or replacing an existing software implementation with custom encryption hardware can be done without any trouble. The stable, worldwide adopted API allows any of the algorithms and modes supported by cryptlib to be used with a minimum of coding effort. In addition the easy-to-use high-level routines allow for the exchange of encrypted or signed messages or the establishment of secure communications channels with a minimum of programming overhead. Language bindings are available for C / C++, C# / .NET, Delphi, Java, Python, and Visual Basic (VB). <http://www.cs.auckland.ac.nz/~pgut001/cryptlib/>

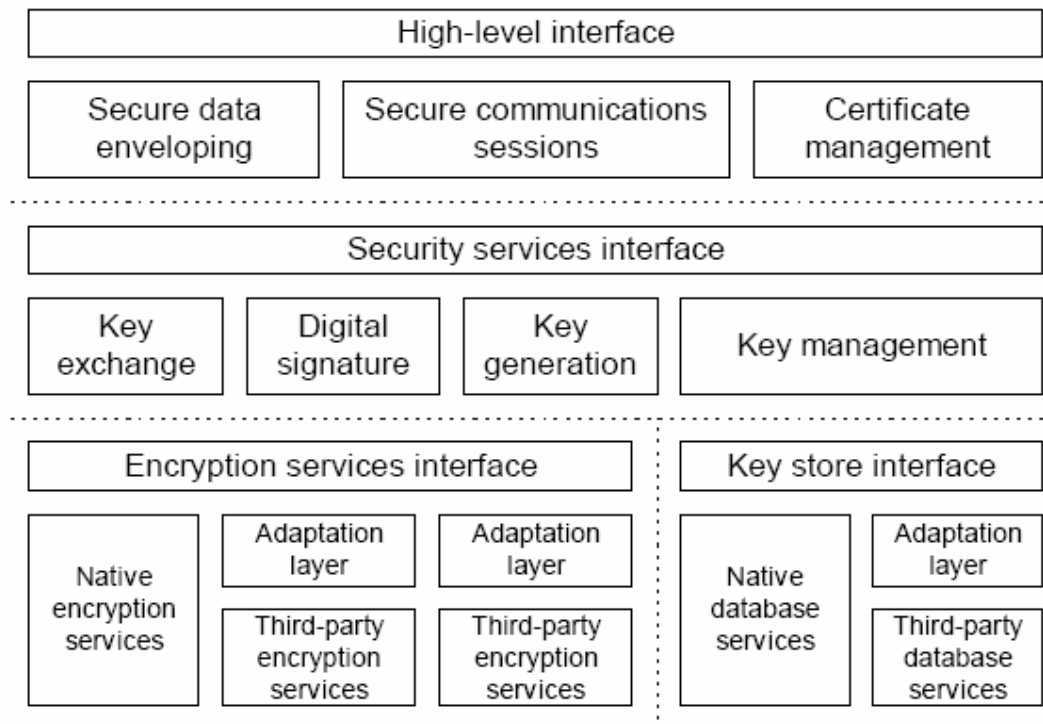
7.2.9.1 cryptlib

cryptlib is a powerful security software library that allows programmers to easily add encryption and authentication services to their software.

DE3.1.2A – Framework and Tools Specification (General and Model)

cryptlib provides a standardised interface to a number of popular encryption algorithms and security services. Cryptlib can handle secure sessions, end-to-end certification management, symmetric and asymmetric key algorithms etc. cryptlib implements many algorithms: AES, DES, DSA, RSA etc

cryptlib has a layered structure. At the top layer the user can find powerful and easy-to-use functions such as “encrypt a message” or “open a secure link”. At the bottom layer there are low level functions such as encryption routines. High level functions make use of the low level functions. The following picture shows the layered structure of cryptlib.



7.2.9.2 How to install Cryptlib on Windows XP

C/C++ programmers have to download a zipped file that contains all the sources from <http://www.cs.auckland.ac.nz/~pgut001/cryptlib/>. Inside the zipped file the user can find all the source files and a workspace file for VisualC++ 6. The result of the compilation with VisualC++ 6 is a library file named **cl32.dll**. The file **cryptlib.h** serves as interface to use **cl32.dll** in your projects.

If you want to use cryptlib in Visual Basic/Visual Basic Script/J++/.Net/Delphi you have to compile cryptlib as described in the above paragraph. Additionally you have to download an ActiveX wrapper named **clcom.dll**. As a side note, I have failed to installing this ActiveX component following the instructions given in the doc.

If you want to use cryptlib in Java you can find compilation instructions in the documentation.

7.2.9.3 How to use cryptlib in C

This example shows how to encrypt a message by using the **cryptEncrypt** function.

Prior to calling the encryption/decryption functions the programmer has to do some initialization work. The following function initializes the cryptlib:

```
cryptInit();
```

Then the programmer has to create a context and choose an encryption algorithm and a key:

```
//context declaration  
CRYPT_CONTEXT cryptContext;
```

```
//Create context an set 3DES algorithm
cryptCreateContext( &cryptContext, cryptUser, CRYPT_ALGO_3DES );

//Set a key
cryptSetAttributeString( cryptContext, CRYPT_CTXINFO_KEY,
"0123456789ABCDEF", 16 );
```

At this point everything is ready to call the encryption/decryption functions.
The following function call encrypts a message stored in the **buffer** variable:

```
cryptEncrypt( cryptContext, buffer, length );
```

cryptContext is a variable that contains information about the encryption algorithm and the key to be used.
buffer is a variable with the message to be encrypted. The encrypted message is returned via the **buffer** and so it overwrites the cleartext.
length is the length of the buffer.

The following function call decrypts the **buffer** according to the algorithm and key provided via **cryptContext**:

```
cryptDecrypt( cryptContext, buffer, length );
```

As you can see from this example cryptlib is not difficult to use. The documentation -300 pages- is available from the cryptlib web page.

7.2.10 Compress/uncompress Support (DSI, lib prob.)

Module Profile		
Compress/Uncompress		
Executable or Library(Support)	Support library	
Single Thread or Multithread		
Language of Development	C/C++	
Responsible Name		
Responsible Partner		
Status (proposed/approved)		
Platforms supported		
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Zlib in wxWidgets	wxWidgets	LGPL

Compress and uncompress support is provided by the wxWidgets library. It provides input stream to access to ZIP files (wxZipInputStream) and input/output streams for gzip compression (wxZlibInputStream, wxZlibOutputStream).

wxZlibInputStream and wxZlibOutputStream are filtering streams they get/send information from/to other wxInput/OutputStream to uncompress/compress information.

For example to compress a buffer of 1024 bytes to a file named “compressed.dat”:

```
char data[1024];

// fill the data buffer

wxFileOutputStream ofile("compressed.dat");
wxZlibOutputStream compress(ofile);

compress.Write(data, 1024);
```

While to uncompress it:

```
wxFileInputStream ifile("compressed.dat");
```

AXMEDIS Project

CONFIDENTIAL

```
wxZlibInputStream uncompress(ifile);  
char data[1024];
```

```
uncompress.Read(data, 1024);  
if(uncompress.LastRead() != 1024)
```

```
...
```

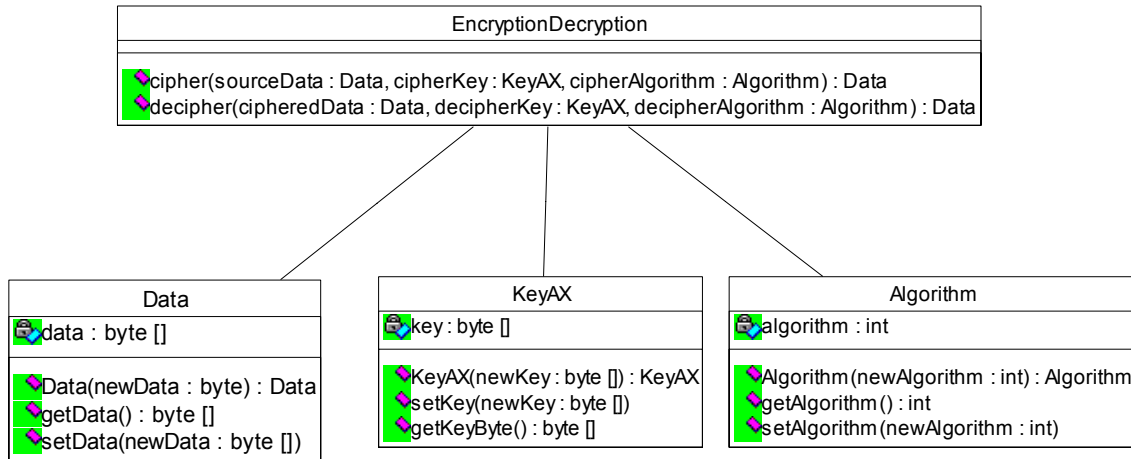
7.2.11 Encryption/Decryption Support (FUPF)

This module provides the needed functionality for encrypting / decrypting AXMEDIS Objects in order to protect them.

Module Profile		
Encryption/Decryption Support		
Executable or Library(Support)	Support library	
Single Thread or Multithread	Single Thread	
Language of Development	C/C++	
Responsible Name		
Responsible Partner	FUPF	
Status (proposed/approved)	Proposed	
Platforms supported	PC (Linux / Windows)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
		Library
File Formats Used	Shared with	File format name or reference to a section
Any		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
N/A		
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
OpenSSL		Apache-style

7.2.11.1 Architecture for encryption / decryption support

Next figure shows the UML description of this module. Nevertheless, this module is based on OpenSSL, so it is not object oriented in nature, but C++ interface could be provided.



Class diagram for the Encryption / Decryption Support

The functionality of the classes inside the UML diagram is as follows:

EncryptionDecryption: Provides the basic functions for ciphering and deciphering of data, hiding to the calling application the complexity derived of the use of the OpenSSL library.

Data: Represents the data (either in clear or ciphered) used by the EncryptionDecryption class.

KeyAX: Represents the key for ciphering / deciphering the data.

Algorithm: Represents the algorithm for ciphering / deciphering data. The list of supported algorithms will be also implemented in this class by using constants in the corresponding programming language (C/C++).

EncryptionDecryption	
Method	cipher
Description	This method ciphers the sourceData passed as parameter using the KeyAX ciphering key and the algorithm indicated by the cipherAlgorithm. The returned information is the ciphered data. This method makes use of the corresponding operations inside the OpenSSL library for the different ciphering algorithms supported by it.
Input parameters	sourceData : Data, the data to be ciphered cipherKey : KeyAX, the key to be used to cipher data cipherAlgorithm : Algorithm, the algorithm used to cipher data
Output parameters	Data, the ciphered data
Method	decipher
Description	This method deciphers the cipheredData passed as parameter using the KeyAX deciphering key and the algorithm indicated by the decipherAlgorithm. The returned information is the data in clear. This method makes use of the corresponding operations inside the OpenSSL library for the different deciphering algorithms supported by it.
Input parameters	cipheredData : Data, the data to be deciphered decipherKey : KeyAX, the key for deciphering the data decipherAlgorithm : Algorithm, the algorithm for the deciphering the data
Output parameters	Data, the original data, in clear

Data	
Method	Data

DE3.1.2A – Framework and Tools Specification (General and Model)

Description	Constructor of the class which receives as parameter the data to be ciphered / deciphered.
Input parameters	NewData: byte[], an array of bytes containing the data either ciphered or in clear
Output parameters	A new instance of the Data class
Method	getData
Description	This method requests the data stored inside this class.
Input parameters	None
Output parameters	byte[], the byte array representing the data contained inside this class
Method	SetData
Description	This method allows setting new data inside this class.
Input parameters	byte[], the byte array representing the data contained inside this class
Output parameters	None

KeyAX	
Method	KeyAX
Description	Constructor of the class which receives as parameter the key for ciphering / deciphering data.
Input parameters	NewKey: byte[], an array of bytes containing the key
Output parameters	A new instance of the KeyAX class
Method	getKey
Description	This method requests the key stored inside this class.
Input parameters	None
Output parameters	byte[], the byte array representing the key contained inside this class
Method	setKey
Description	This method allows setting new key inside this class.
Input parameters	byte[], the byte array representing the key contained inside this class
Output parameters	None

Algorithm	
Method	Algorithm
Description	Constructor of the class which receives as parameter the algorithm identifier
Input parameters	NewAlgorithm:int, the identifier of the algorithm contained inside this class. It will depend on the values
Output parameters	A new instance of the Algorithm class
Method	getAlgorithm
Description	This method requests the algorithm stored inside this class.
Input parameters	None
Output parameters	int, the identifier of the algorithm contained inside this class. It will depend on the values defined by OpenSSL
Method	setAlgorithm

DE3.1.2A – Framework and Tools Specification (General and Model)

Description	This method allows setting new algorithm inside this class.
Input parameters	int, the identifier of the algorithm contained inside this class. It will depend on the values defined by OpenSSL
Output parameters	None

Application Programming Interface

The API of OpenSSL serves as an interface to a range of security functions. In this module we will use the ones that provide cryptographic functionality. In next section, OpenSSL library is explained briefly.

7.2.11.2 OpenSSL

The OpenSSL Project (<http://www.openssl.org>) is a collaborative effort to develop a robust, commercial-grade, full-featured and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan and develop the OpenSSL toolkit and its related documentation.

OpenSSL is based on the SSLeay library developed by Eric A. Young and Tim J. Hudson. The OpenSSL toolkit is licensed under an Apache-style licence, which basically means that you are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions.

7.2.11.3 Windows Version of OpenSSL

The Windows version of OpenSSL library can be downloaded from <http://www.slproweb.com/products/Win32OpenSSL.html>

7.2.11.4 Cryptographic functions provided by OpenSSL

The OpenSSL crypto library implements a wide range of cryptographic algorithms used in various Internet standards. The services provided by this library are used by the OpenSSL implementations of SSL, TLS and S/MIME, and they have also been used to implement SSH, OpenPGP, and other cryptographic standards.

libcrypto consists of a number of sub-libraries that implement the individual algorithms. The functionality includes symmetric encryption, public key cryptography and key agreement, certificate handling, cryptographic hash functions and a cryptographic pseudo-random number generator. It is briefly described next.

- Symmetric ciphers: des, idea, rc2, rc4, rc5.
- Public key cryptography and key agreement: dsa, dh, rsa.
- Certificates: x509, x509v3.
- Authentication codes, hash functions: hmac, md4, md5, sha.
- Input/Output data encoding: asn1, bio, evp, pem, pkcs7, pkcs12.

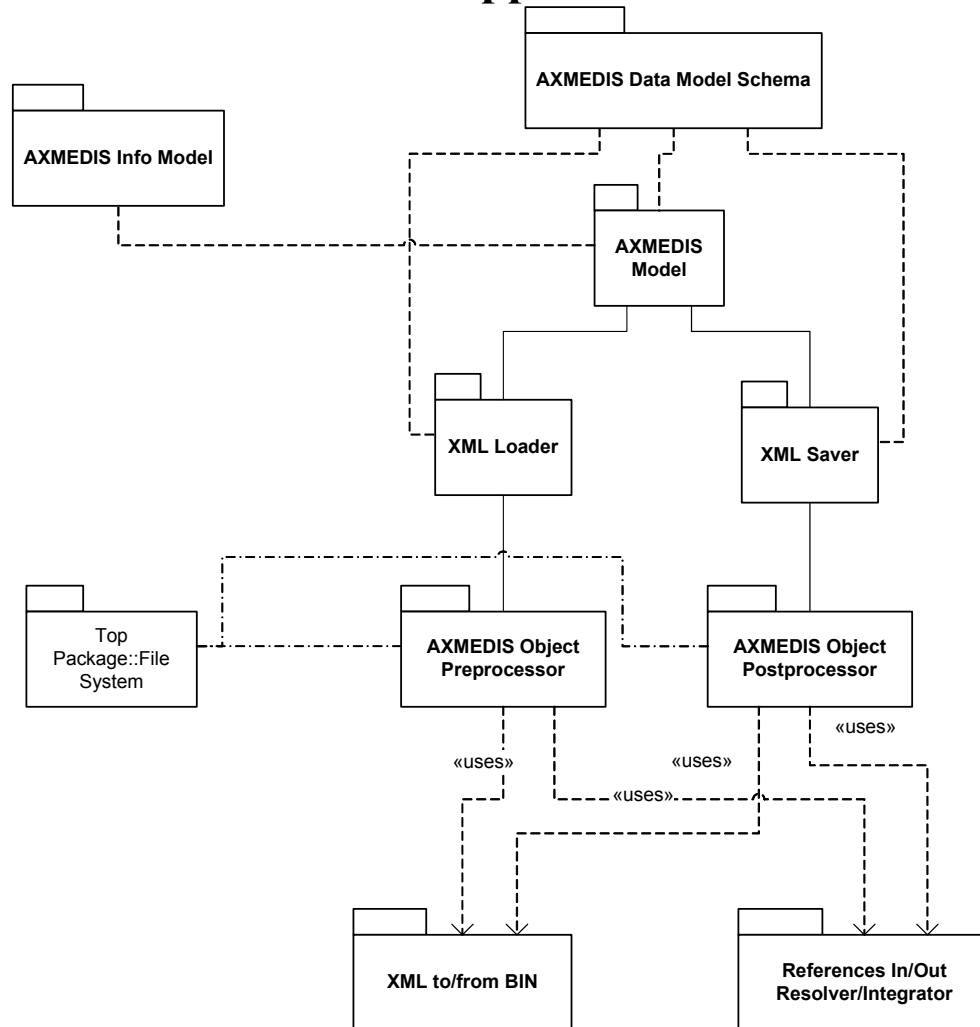
The functions provided by each algorithm depend on its nature, and are independently described in the library documentation (for details, see <http://www.openssl.org/docs/crypto/crypto.html>).

8 AXMEDIS Data Model Support (WP4.1.2: DSI, WP5.4.3: DSI, EPFL)

It is comprised of:

- **AXMEDIS Data Model Schema:** schema of the XML model for AXMEDIS objects. It also includes a description of the AXMEDIS details that make of AXMEDIS Objects a specialization of MPEG21 Digital Items.
- **AXMEDIS Model:** classes modeling the AXMEDIS objects
- **XML Loader and Saver:** classes for XML loading and saving
- **AXMEDIS Object Preprocessor and Postprocessor:** Tools for transforming XML into binary and viceversa, for transforming external references in internal and viceversa, etc.

AXMEDIS Data Model Support



Module Profile	
AXMEDIS DataModel Support	
Executable or Library(Support)	Support library
Single Thread or Multithread	
Language of Development	C++
Responsible Name	
Responsible Partner	DSI
Status (proposed/approved)	proposed

DE3.1.2A – Framework and Tools Specification (General and Model)

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

8.1 AXMEDIS Data Model Schema (DSI)

It also includes a description of the AXMEDIS details that make of AXMEDIS of a specialization of MPEG21 format.

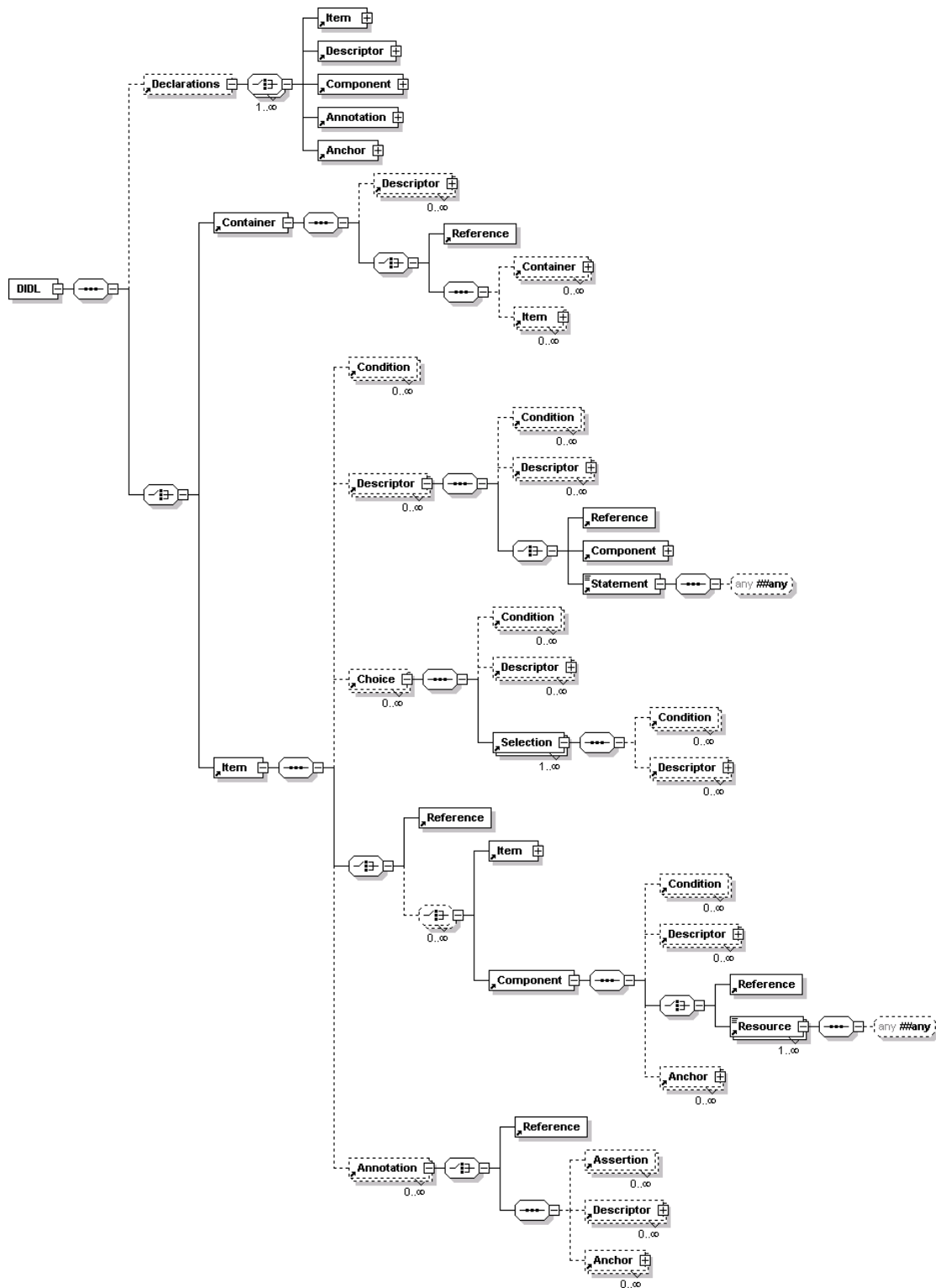
8.1.1 AXMEDIS Objects as MPEG21 Objects

An AXMEDIS Object has to be an MPEG21 digital item but any MPEG21 digital item is not an AXMEDIS Object. This means that an AXMEDIS Object will have a specific structure and will not support all the extremely flexible structuring capabilities of MPEG21 digital items.

In this section will be investigated how AXMEDIS Objects could be represented using the structuring features of MPEG21.

8.1.1.1 MPEG21 Digital Items

The following figure describes how a MPEG21 Digital Item is structured:



for a complete description see part 2 of the MPEG21 standard describing the Digital Item Description Language. This part of the standard is related to unprotected digital items only.

The elements contained in a MPEG21 digital items are:

- **Container** – is a container of items or of other containers;

DE3.1.2A – Framework and Tools Specification (General and Model)

- **Item** – represents a digital item, it contains *Descriptors* (metadata of the whole digital item) *Components* (content that builds up the item) directly or through *Reference*, and it also contains other secondary elements;
- **Descriptor** – contains metadata through a *Statement* element or a *Component* (e.g. for thumbnails)
- **Component** – contains *Resources* and *Descriptors* (metadata of the resource);
- **Resource** – contains an external reference to the resource (audio, video, text,...) or it can host it inside the element using base64 encoding;
- **Reference** – is a place holder for another element it refers to;
- **Annotation** – contains an annotation
- **Anchor** – is a link into the content
- **Condition, Choice, Selection** – are used to group sub parts of the item on the basis of end user selections, this to avoid streaming of big items.

When considering **protected** MPEG21 digital items, the standardization process of this feature, is not currently at level of International Standard but at level of Committee Draft.

Protected content in MPEG21 is obtained by substituting a sub tree of the original XML tree with an element having the same name (but with different namespace) and containing the protected version of the sub tree in binary form and the additional information needed to enable access to the content.

8.1.1.2 AXMEDIS Objects

AXMEDIS Objects can be classified as:

Basic AXMEDIS Object: containing one or more digital resources (image, video, document, etc.) and metadata related to the whole object. Resources can be stored inside the object or outside.

Protected Basic AXMEDIS Object: containing one or more protected digital resources and metadata of the whole object (in clear but certified). Protected resources can be stored inside the object or outside.

Composite AXMEDIS Object: containing a set of AXMEDIS Objects (Basic or Composite, protected or not). It has specific metadata for the whole object in addition to metadata related to sub-objects. The sub-objects can be stored inside the object or referenced.

Protected Composite AXMEDIS Object: as the previous but the whole object is protected (the metadata of the whole object and of the sub-objects has to be accessible in clear)

Governed AXMEDIS Object: anyone of the previous containing the license to use the object

Query/Promotional AXMEDIS Object: containing only metadata of an object and reference to the real content.

In the following possible mappings of AXMEDIS Objects as MPEG21 digital items are reported. A tree like structure is used to represent the XML structure.

MPEG21 *Descriptors* are used to contain metadata related to the content. The *Statement* element inside the *Descriptor* can contain any XML or text, MPEG21 does not fix its content.

The order where *Descriptor* elements are reported is not fixed, however some of them are required (have to be present) and others are optional (may be missing). Some *Descriptors* are specified in the standard for Digital Item Identification:

- *Identifier*, used to identify the object, MPEG21 does not provide a new identification scheme but it allows to host any kind of identification code. A Registration Authority will be set up to register identification schemes to be used in MPEG21 Digital Items. A URI is used as identifier, for Example: `<dii:Identifier>urn:mpegRA:mpeg21:dii:isrc:US-ZO3-99-32476</dii:Identifier>` identifies an object using a ISRC code. An Identifier can be used to store the AXMEDIS Object ID.
- *RelatedIdentifier*, used to identify the work with a uri. It can be used to store the AXMEDIS Work ID. Example: `<dii:RelatedIdentifier>urn:mpegRA:mpeg21:dii:iscw:T-034.524.680-1</dii:RelatedIdentifier>`

The *AXInfo* element is used to contain information specific for AXMEDIS framework. Metadata like title, author, etc. and mpeg7 metadata are not stored inside *AXInfo* to allow MPEG21 terminals to access to these

DE3.1.2A – Framework and Tools Specification (General and Model)

metadata even if they are not AXMEDIS compliant tools. Other AXMEDIS specific metadata related to the content can be defined (e.g. for technical information), and hosted in specific *Descriptor* elements, if a suitable standardized format is not available (e.g. mpeg7).

Since all metadata could be accessible also for protected objects, an AXMEDIS object is structured in the following way:

DIDL

Item

Descriptor containing AXOID

Item

contains structured metadata of the object (always in clear) and reference to the content

Item id="AXOID1_content"

contains the real object, it contains both metadata and content, this Item could be protected or not

The following example shows an example of an object with multiple descriptors:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS" xmlns:ax="urn:axmedis:01" xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS" xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:axmedis:01 AXMEDIS.xsd">
  <Item>
    <!--
      Descriptor containing the AXOID to identify the object (REQUIRED)
    -->
    <Descriptor id="dsc_id">
      <Statement mimeType="text/xml">
        <dii:Identifier>urn:mpegRA:mpeg21:dii:axoid:A001AGSHDI</dii:Identifier>
      </Statement>
    </Descriptor>
  </Item>
  <!--
    Descriptor containing the AXInfo containing information regarding the object (REQUIRED)
  -->
  <Descriptor id="public_dsc_ax">
    <Statement mimeType="text/xml">
      <ax:AXInfo>
        ...
      </ax:AXInfo>
    </Statement>
  </Descriptor>
  <!--
    Descriptor containing the Dublin Core information regarding the object (REQUIRED)
  -->
  <Descriptor id="public_dsc_dc">
    <Statement mimeType="text/xml">
      <rdf:Description>
        <dc:title xml:lang="en">When the Thistle Blooms</dc:title>
        <dc:title xml:lang="it">Quando il Cardo Sboccia</dc:title>
        <dc:creator>Always Red</dc:creator>
        <dc:publisher>PDQ Records</dc:publisher>
      </rdf:Description>
    </Statement>
  </Descriptor>
  <!--
    Descriptor containing the MPEG7 information regarding the object (OPTIONAL)
  -->
  <Descriptor id="public_dsc_mpeg7">
    <Statement mimeType="text/xml">
      <mpeg7:Mpeg7>
        ...
      </mpeg7:Mpeg7>
    </Statement>
  </Descriptor>
  <Descriptor>
    <Descriptor><Statement mimeType="text/plain">Reference to the real content</Statement></Descriptor>
    <Statement mimeType="text/uri-list">urn:axmedis:A001AGSHDI#A001AGSHDI_content</Statement>
  </Descriptor>
</DIDL>
```

```

    </Descriptor>
  </Item>
  <Item id="A001AGSHDI_content">
    <!--
      Descriptor containing the AXInfo containing information regarding the object (REQUIRED)
    -->
    <Descriptor id="private_dsc_ax">
      <Statement mimeType="text/xml">
        <ax:AXInfo>
          ...
        </ax:AXInfo>
      </Statement>
    </Descriptor>
    <!--
      Descriptor containing the Dublin Core information regarding the object (REQUIRED)
    -->
    <Descriptor id="private_dsc_dc">
      <Statement mimeType="text/xml">
        <rdf:Description>
          <dc:title xml:lang="en">When the Thistle Blooms</dc:title>
          <dc:title xml:lang="it">Quando il Cardo Sboccia</dc:title>
          <dc:creator>Always Red</dc:creator>
          <dc:publisher>PDQ Records</dc:publisher>
        </rdf:Description>
      </Statement>
    </Descriptor>
    <!--
      Descriptor containing the MPEG7 information regarding the object (OPTIONAL)
    -->
    <Descriptor id="private_dsc_mpeg7">
      <Statement mimeType="text/xml">
        <mpeg7:Mpeg7>
          ...
        </mpeg7:Mpeg7>
      </Statement>
    </Descriptor>
    <!--
      Component elements containing the resource (REQUIRED for single object)
    -->
    <Component id="cmp">
      <Resource mimeType="video/mp4v-es" encoding="base64">
        aadsfadsfsyd647dgd78r85hfuv8nbr8fnf985nf9g9gm569gnty9ghmg90hdhd8fhjd9d9
        dhd8f95mnfk9gfm59fgt95mkt0jhdf8fnj587fjd67n3jf84mf00eedjf8fj58tm58fm58emds9o
        ...
      </Resource>
    </Component>
  </Item>
</Item>
</DIDL>

```

References in an object to other objects can be done using the AXOID. This allows to reconstruct objects relations in any other place. An additional complexity is due to the use of temporary AXOIDs which are forbidden to be used outside the AXMEDIS Factory.

References to resources (audio, document, video, ... files) can be done using a path. However have to be noted that a resource have not a unique ID this means that sharing a resource among objects is not possible.

8.1.1.3 Basic AXMEDIS Object:

DIDL
Item
Descriptor
Statement
dii:Identifier (contains the AXOID, REQUIRED)
urn:mpegRA:mpeg21:dii:axoid:AXOID1
Item
Descriptor
Statement

DE3.1.2A – Framework and Tools Specification (General and Model)

dii:Identifier	(contains any other identifier, OPTIONAL)
Descriptor	
Statement	
dii:RelatedIdentifier	(contains the WorkID, OPTIONAL)
Descriptor	
Statement	
dii:Type	(contains the type of object, OPTIONAL)
Descriptor	
Statement	
ax:AXInfo	(contains AXMEDIS specific information, REQUIRED)
Descriptor	
Statement	
rdf:Description	(contains Dublin Core metadata, REQUIRED)
Descriptor	
Statement	
mpeg7:Mpeg7	(contains MPEG7 metadata, OPTIONAL)
Descriptor	
Statement	
???:XXX	(contains any other metadata in XML, OPTIONAL)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID1#AXOID1_content	
Item id="AXOID1_content"	
Descriptor	
Statement	
dii:Identifier	(contains any other identifier, OPTIONAL)
Descriptor	
Statement	
dii:RelatedIdentifier	(contains the WorkID, OPTIONAL)
Descriptor	
Statement	
dii:Type	(contains the type of object, OPTIONAL)
Descriptor	
Statement	
ax:AXInfo	(contains AXMEDIS specific information, REQUIRED)
Descriptor	
Statement	
rdf:Description	(contains Dublin Core metadata, REQUIRED)
Descriptor	
Statement	
mpeg7:Mpeg7	(contains MPEG7 metadata, OPTIONAL)
Descriptor	
Statement	
???:XXX	(contains any other metadata in XML, OPTIONAL)
Component	
Resource	(contains/refers the resource, REQUIRED)
Component	(another component, OPTIONAL)
Resource	(contains/refers the resource, REQUIRED)
...	

Thus a *Basic AXMEDIS Object* is structured in the following way:

DIDL	
Item	
OBJECT_AXOID	
Item	(contains the metadata of the object and of contained objects)
OBJECT_METADATA	
CONTENT_REFERENCE	(contains an URI referring to the real content, REQUIRED)
Item id="AXOID_content"	
OBJECT_METADATA	
CONTENT	

where:

- **OBJECT_AXOID** is a Descriptor containing the AXOID of the basic object;

DE3.1.2A – Framework and Tools Specification (General and Model)

- *OBJECT_METADATA* is a sequence of Descriptors containing the metadata of the basic object;
- *CONTENT_REFERENCE* is a descriptor containing the reference to the content;
- *CONTENT* is a sequence of Components

Note: Multiple components will be used for HTML documents containing images inside. Normally only one component is present.

8.1.1.4 Protected Basic AXMEDIS Object:

A Protected AXMEDIS Object is obtained by protecting the Item containing the real content.

DIDL	
Item	
Descriptor	
Statement	
dii:Identifier	(contains the AXOID, REQUIRED)
urn:mpegRA:mpeg21:dii:axoid:AXOID1	
Item	
Descriptor	
Statement	
dii:Identifier	(contains any other identifier, OPTIONAL)
Descriptor	
Statement	
dii:RelatedIdentifier	(contains the WorkID, OPTIONAL)
Descriptor	
Statement	
dii:Type	(contains the type of object, OPTIONAL)
Descriptor	
Statement	
ax:AXInfo	(contains AXMEDIS specific information, REQUIRED)
Descriptor	
Statement	
rdf:Description	(contains Dublin Core metadata, REQUIRED)
Descriptor	
Statement	
mpeg7:Mpeg7	(contains MPEG7 metadata, OPTIONAL)
Descriptor	
Statement	
???:XXX	(contains any other metadata in XML, OPTIONAL)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID1#AXOID1_content	
ipmpdidl:Item	(contains/refers the protected components, REQUIRED)
ipmpdidl:Identifier id="AXOID1_content"	
ipmpdidl:Content	
XXXXXXXXXXXXXXXXXXXXX	
...	
XXXXXXXXXXXXXXXXXXXXX	

8.1.1.5 Composite AXMEDIS Object:

A composite object obtained by composing objects $O_1, O_2, \dots O_n$ is structured as follows:

DIDL	
Item	
<i>OBJECT AXOID</i>	
Item	(contains the metadata of the object)
<i>OBJECT_METADATA</i>	
<i>CONTENT_REFERENCE</i>	
(contains an URI referring to the real content, REQUIRED)	
METADATA[O_1]	
METADATA[O_2]	
...	
METADATA[O_n]	
Item id="AXOID_content"	(contains the content of the object)

<i>OBJECT_METADATA</i>
FIRST_ITEM[O_1]
FIRST_ITEM[O_2]
...
FIRST_ITEM[O_n]

where:

- *OBJECT_AXOID* is a Descriptor containing the AXOID of the composed object;
- *OBJECT_METADATA* is a sequence of Descriptors containing the metadata of the whole object
- *CONTENT_REFERENCE* is a descriptor containing the reference to the content
- METADATA[O] is a function used to extract from the object O the sub-tree containing only the metadata of the object
- FIRST_ITEM[O] is a function to get the first child item of the object, it is used to skip the DIDL tag.

the following is an example of double composition:

AXOID1 = COMPOSE(AXOID2, AXOID3 = COMPOSE(AXOID4, AXOID5))

AXOID1 – a composite object

AXOID2 – a basic object

AXOID3 – a composite object

AXOID4 – a basic object

AXOID5 – a basic object

DIDL	
Item	
Descriptor	(contains AXOID1)
Item	(contains the metadata of the object)
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID1#AXOID1_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID2#AXOID2_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID3#AXOID3_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID4#AXOID4_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID5#AXOID5_content	
Item id="AXOID1_content"	(contains the metadata and the content)
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Item	

DE3.1.2A – Framework and Tools Specification (General and Model)

Descriptor	(contains AXOID2)
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID2#AXOID2_content	
Item id="AXOID2_content"	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Component	
Item	
Descriptor	(contains AXOID3)
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID3#AXOID3_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to AXOID4_content)
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to AXOID5_content)
Item id="AXOID3_content"	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Item	
Descriptor	(contains AXOID4)
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to AXOID4_content)
Item id="AXOID4_content"	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Component	
Item	
Descriptor	(contains AXOID5)
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to AXOID5_content)
Item id="AXOID5_content"	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Component	

8.1.1.6 Protected Composite AXMEDIS Object:

A Protected Composite AXMEDIS Object is obtained, as for basic objects, protecting the Item containing the real content.

DIDL	
Item	
Descriptor	(contains AXOID1)
Item	(contains the metadata of the object)
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID1#AXOID1_content	

Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID2#AXOID2_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID3#AXOID3_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID4#AXOID4_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID5#AXOID5_content	
ipmpdidl:Item	(contains/refers the protected components, REQUIRED)
ipmpdidl:Identifier id="AXOID1_content"	
ipmpdidl:Identifier id="AXOID2_content"	
ipmpdidl:Identifier id="AXOID3_content"	
ipmpdidl:Identifier id="AXOID4_content"	
ipmpdidl:Identifier id="AXOID5_content"	
ipmpdidl:Content	
XXXXXXXXXXXXXXXXXXXX	
...	
XXXXXXXXXXXXXXXXXXXX	

8.1.1.7 Governed AXMEDIS Object:

A Governed AXMEDIS Object contains the licence inside a descriptor like in the following example:

DIDL	
Item	
Descriptor	
Statement	
dii:Identifier	(contains the AXOID, REQUIRED)
Item	
Descriptor	
Statement	
dii:RelatedIdentifier	(contains the WorkID, OPTIONAL)
Descriptor	
Statement	
ax:AXInfo	(contains AXMEDIS specific information, REQUIRED)
Descriptor	
Statement	
rdf:Description	(contains Dublin Core metadata, REQUIRED)
Descriptor	
Statement	
mpeg7:Mpeg7	(contains MPEG7 metadata, OPTIONAL)
Descriptor	
Statement	
r:license	(contains the license for the object, OPTIONAL)
...	(any other of the previous structures)

8.1.1.8 Query/Promotional AXMEDIS Object

A Query/Promotional object contains only metadata and it references to the real content. The following is an example.

DIDL	
Item	
Descriptor	(contains AXOID1)
Item	(contains the metadata of the object)
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID1#AXOID1_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID2#AXOID2_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID3#AXOID3_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID4#AXOID4_content	
Item	
Descriptor	(contains AXinfo)
Descriptor	(contains Dublin Core metadata)
Descriptor	(contains an URI referring to the real content, REQUIRED)
Statement mimeType="text/uri-list"	
urn:axmedis:AXOID5#AXOID5_content	

8.1.2 AXMEDIS Metadata Model (DSI, EPFL,)

Metadata information related to an object, as seen in previous section, is split among various Descriptors.

These Descriptors can contain:

- identification information (as standardized by MPEG21) for AXOID, AXWID and Type
- AXMEDIS specific information regarding object life-cycle (in AXInfo)
- Dublin Core metadata
- MPEG 7 metadata
- any other metadata represented in XML

8.1.2.1 AXInfo Model

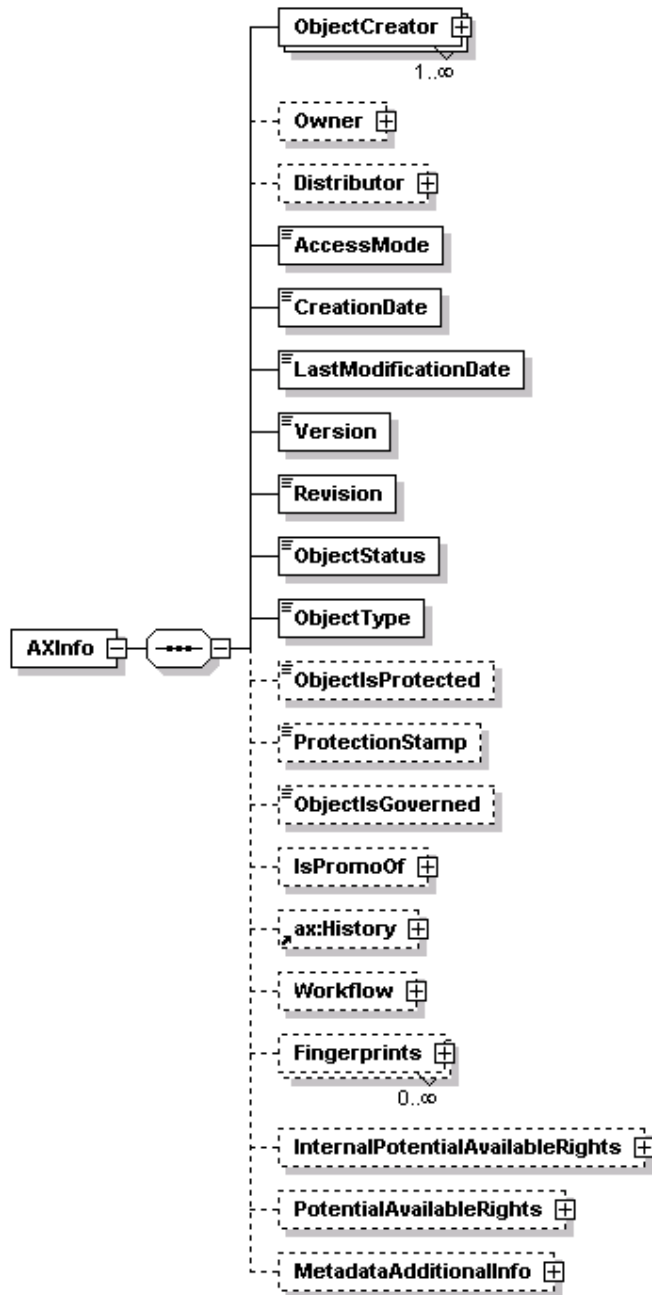
The AXInfo contains information to manage the object in its entire life-cycle, it contains

- Creator information (AXCID, Name, Company, URLs, ...)
- Distributor information (AXDID, Name, URLs, ...)
- Access information (read only or read/write)
- Creation and modification times
- The History of the object (version/revision, commands performed on the object)
- The Workflow information, etc.
- Fingerprinting information (algorithm identification)
- Potential Available Rights (PAR) for the object and licensing information
- Metadata certification information used to check metadata consistency

In the following documentation of AXInfo schema is reported.

DE3.1.2A – Framework and Tools Specification (General and Model)
 element **AXInfo**

diagram



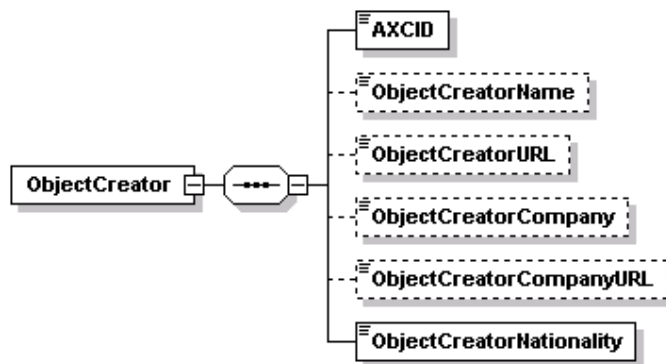
namespace urn:axmedis:01

children [ObjectCreator](#) [Owner](#) [Distributor](#) [AccessMode](#) [CreationDate](#) [LastModificationDate](#) [Version](#) [Revision](#) [ObjectStatus](#) [ObjectType](#) [ObjectIsProtected](#) [ProtectionStamp](#) [ObjectIsGoverned](#) [IsPromoOf](#) [ax:History](#) [Workflow](#) [Fingerprints](#) [InternalPotentialAvailableRights](#) [PotentialAvailableRights](#) [MetadataAdditionalInfo](#)

description

element **AXInfo/ObjectCreator**

diagram



namespace urn:axmedis:01

children [AXCID](#) [ObjectCreatorName](#) [ObjectCreatorURL](#) [ObjectCreatorCompany](#) [ObjectCreatorCompanyURL](#) [ObjectCreatorNationality](#)

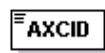
description It contains information regarding the person who created the object

example

```
<ObjectCreator>
  <AXCID>9383726716152748549594873723</AXCID>
  <ObjectCreatorName>John Doe</ObjectCreatorName>
  <ObjectCreatorURL>mailto:j.doe@video2.org</ObjectCreatorURL>
  <ObjectCreatorCompany>VIDEO2</ObjectCreatorCompany>
  <ObjectCreatorCompanyURL>http://www.video2.com</ObjectCreatorCompanyURL>
  <ObjectCreatorNationality>US</ObjectCreatorNationality>
</ObjectCreator>
```

element **AXInfo/ObjectCreator/AXCID**

diagram



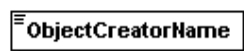
namespace urn:axmedis:01

type **xs:string**

description It contains the AXMEDIS Creator Identifier

element **AXInfo/ObjectCreator/ObjectCreatorName**

diagram



namespace urn:axmedis:01

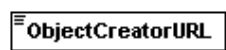
type **xs:string**

description personal name of the creator

constraints This tag should be removed when published on the P2P or on B2C

element **AXInfo/ObjectCreator/ObjectCreatorURL**

diagram



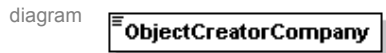
namespace urn:axmedis:01

type **xs:anyURI**

description URL associated to the object creator, it could be the email address

constraints This tag should be removed when published on the P2P or on B2C

element **AXInfo/ObjectCreator/ObjectCreatorCompany**

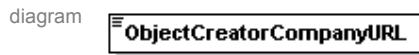


namespace urn:axmedis:01

type **xs:string**

description name of the company of the creator

element **AXInfo/ObjectCreator/ObjectCreatorCompanyURL**



namespace urn:axmedis:01

type **xs:anyURI**

description URL of the company of the creator

element **AXInfo/ObjectCreator/ObjectCreatorNationality**

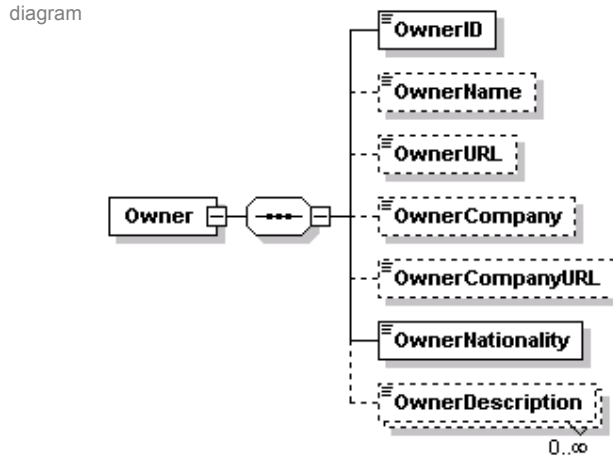


namespace urn:axmedis:01

type **xs:string**

description nationality of the creator company using the ISO 3166 two letters code

element **AXInfo/Owner**



namespace urn:axmedis:01

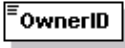
children [OwnerID](#) [OwnerName](#) [OwnerURL](#) [OwnerCompany](#) [OwnerCompanyURL](#) [OwnerNationality](#) [OwnerDescription](#)

description It contains information regarding the owner of the content, if not present the creator is the owner

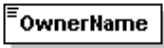
example

```
<Owner>
  <OwnerID coding="SIAE">0038367292-292893-202383</OwnerID>
  <OwnerCompany>VIDEO Production</OwnerCompany>
  <OwnerCompanyURL>http://www.videoproduction.com</OwnerCompanyURL>
  <OwnerNationality>US</OwnerNationality>
</Owner>
```

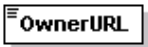
element **AXInfo/Owner/OwnerID**

diagram					
namespace	urn:axmedis:01				
type	extension of xs:string				
attributes	Name	Type	Use	Default	Fixed
	coding	xs:string	required		
description	identification code to identify the content owner, the coding attribute is used to state which coding scheme is used				
example	<OwnerID coding="SIAE">10293834-236272-353</OwnerID>				


element **AXInfo/Owner/OwnerName**

diagram	
namespace	urn:axmedis:01
type	xs:string
description	name of the content owner

element **AXInfo/Owner/OwnerURL**

diagram	
namespace	urn:axmedis:01
type	xs:anyURI
description	the URL of the owner (website or email)

element **AXInfo/Owner/OwnerCompany**

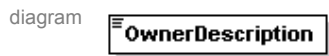
diagram	
namespace	urn:axmedis:01
type	xs:string
description	company name owning the content

element **AXInfo/Owner/OwnerCompanyURL**

diagram	
namespace	urn:axmedis:01
type	xs:string
description	URL of the company owning the content (web site)

element **AXInfo/Owner/OwnerNationality**

diagram	
namespace	urn:axmedis:01
type	xs:string
description	Nationality of the content owner encoded using ISO 3166 two letters code

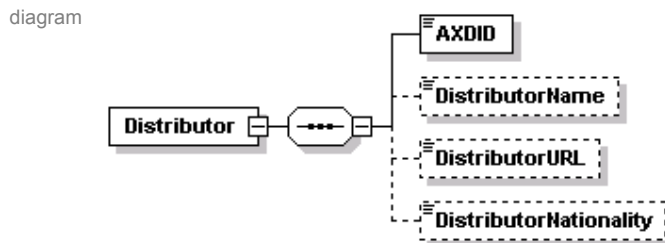
element AXInfo/Owner/OwnerDescription

namespace urn:axmedis:01

type extension of **xs:string**

attributes	Name	Type	Use	Default	Fixed
	lang	xs:string	required		

description A description of the owner, the *lang* attribute states the language used for the description

element AXInfo/Distributor

namespace urn:axmedis:01

children [AXDID](#) [DistributorName](#) [DistributorURL](#) [DistributorNationality](#)

description It contains information about the Distributor that distributed the object, it will be present only in the B2C phase

element AXInfo/Distributor/AXDID

namespace urn:axmedis:01

type **xs:string**

description is the AXMEDIS Distributor Identifier

element AXInfo/Distributor/DistributorName

namespace urn:axmedis:01

type **xs:string**

description name of the distributor

element AXInfo/Distributor/DistributorURL

namespace urn:axmedis:01

type **xs:anyURI**

description URL of the distributor (web site)

element AXInfo/Distributor/DistributorNationality

namespace urn:axmedis:01

DE3.1.2A – Framework and Tools Specification (General and Model)

type **xs:string**

description Nationality of the distributor encoded using ISO 3166 two letters code

element **AXInfo/AccessMode**



namespace urn:axmedis:01

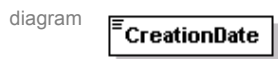
type restriction of **xs:string**

facets enumeration readOnly
enumeration read_write

description states if the object can be changed (read_write) or not (readOnly)

constraints The AccessMode should be the same in all the AXInfos of a composite object, however in case they are missing or contradictory the one at the top level should be considered valid.

element **AXInfo/CreationDate**

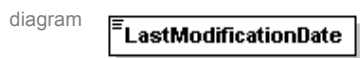


namespace urn:axmedis:01

type **xs:dateTime**

description date and time of object creation

element **AXInfo/LastModificationDate**

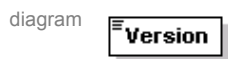


namespace urn:axmedis:01

type **xs:dateTime**

description date and time of object modification

element **AXInfo/Version**



namespace urn:axmedis:01

type **xs:nonNegativeInteger**

description number of version of the object, it should be incremented each time the object is uploaded in the AXDB

element **AXInfo/Revision**



namespace urn:axmedis:01

type **xs:nonNegativeInteger**

description number of revision of the object, it should be incremented each time the object is saved to disk and it should return to 0 when uploaded in the AXDB

element **AXInfo/ObjectStatus**



namespace urn:axmedis:01

DE3.1.2A – Framework and Tools Specification (General and Model)

type **xs:string**

description status of the object (e.g. in production, published, ...)

element **AXInfo/ObjectType**



namespace urn:axmedis:01

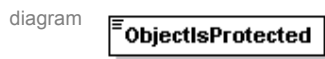
type restriction of **xs:string**

facets
enumeration BASIC
enumeration COMPOSITE

description it states if the object is BASIC or COMPOSITE

constraints in case the object is BASIC it should have the structure of an AXMEDIS Basic Object, and the structure of an AXMEDIS Composite Object for a COMPOSITE one. The value for this tag can be also derived from the object structure.

element **AXInfo/ObjectsIsProtected**

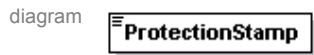


namespace urn:axmedis:01

type **xs:boolean**

description states if the object is protected (true) or not (false)

element **AXInfo/ProtectionStamp**



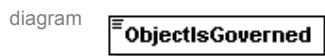
namespace urn:axmedis:01

type **xs:string**

description a protection stamp identifying the protected object

constraints It has to be present in case of protected object

element **AXInfo/ObjectsIsGoverned**

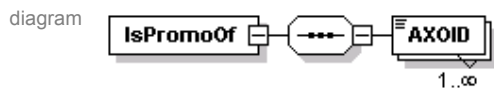


namespace urn:axmedis:01

type **xs:boolean**

description states if the object has a licence inside (true) or not (false)/

element **AXInfo/IsPromoOf**



namespace urn:axmedis:01

children [AXOID](#)

description contains a sequence of AXOIDs referring to objects for which this object is a promotional version

element **AXInfo/IsPromoOf/AXOID**



DE3.1.2A – Framework and Tools Specification (General and Model)

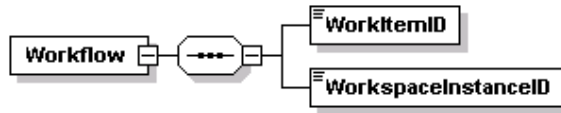
namespace urn:axmedis:01

type **xs:token**

description an identifier of an AXMEDIS object for which the whole object is a promotional version

element **AXInfo/Workflow**

diagram



namespace urn:axmedis:01

children [WorkItemID](#) [WorkspaceInstanceID](#)

description it contains information for the workflow management of the object

element **AXInfo/Workflow/WorkItemID**

diagram



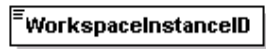
namespace urn:axmedis:01

type **xs:string**

description the identifier of the workitem

element **AXInfo/Workflow/WorkspaceInstanceID**

diagram



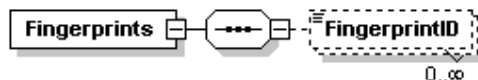
namespace urn:axmedis:01

type **xs:string**

description Identifies the workspace instance

element **AXInfo/Fingerprints**

diagram



namespace urn:axmedis:01

children [FingerprintID](#)

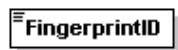
attributes	Name for	Type	Use	Default	Fixed
		xs:IDREF	required		

description contains information regarding fingerprints for a specific Component element of the object.

constraint the for attribute refers to a Component element containing the resource fingerprinted

element **AXInfo/Fingerprints/FingerprintID**

diagram



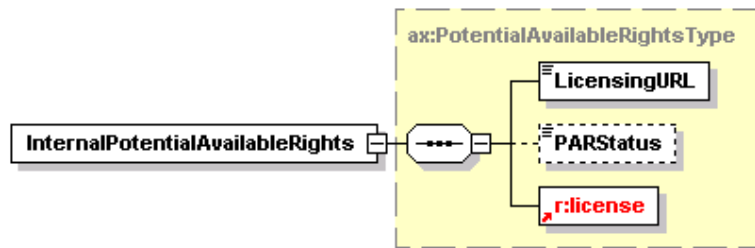
namespace urn:axmedis:01

type extension of **xs:string**

description an identifier for the algorithm used to fingerprint the component

element **AXInfo/InternalPotentialAvailableRights**

diagram



namespace urn:axmedis:01

type [ax:PotentialAvailableRightsType](#)

children [LicensingURL](#) [PARStatus](#) r:license

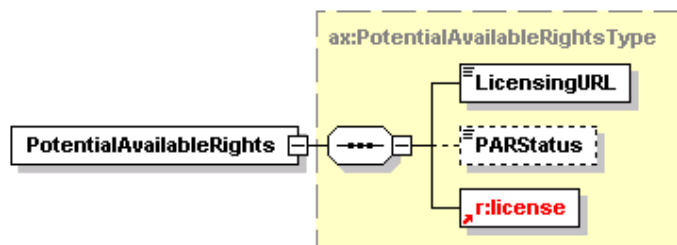
description contains all the rights really available on the object, not all these rights can be exploited by end users or distributors.

the LicensingURL contains the URL to acquire a license for the object and PARStatus contains the status like to be verified, verified, ...

constraints the license is not a complete licence, it is used to contain only the grants but without the principal and the resource elements.

element **AXInfo/PotentialAvailableRights**

diagram



namespace urn:axmedis:01

type [ax:PotentialAvailableRightsType](#)

children [LicensingURL](#) [PARStatus](#) r:license

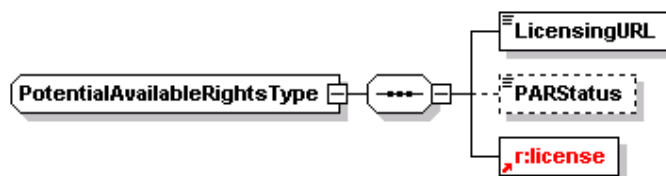
description contains the rights available outside the AXMEDIS Factory usually it is a subset of the InternalPotentialAvailableRights.

the LicensingURL contains the URL to acquire a license for the object and PARStatus contains the status like to be verified, verified, ...

constraints the license is not a complete licence, it is used to contain only the grants but without the principal and the resource elements.

complexType **PotentialAvailableRightsType**

diagram



namespace urn:axmedis:01

children [LicensingURL](#) [PARStatus](#) r:license

description this type contains the information on the rights potentially available on the object, its status and the url to be used to acquire a real license

element **PotentialAvailableRightsType/LicensingURL**



namespace urn:axmedis:01

type **xs:anyURI**

description contains the URL to be used to acquire a licence for the object

element **PotentialAvailableRightsType/PARStatus**



namespace urn:axmedis:01

type **xs:string**

description contains the current status of the PAR like: to be verified, verified, ...

element **AXInfo/MetadataAdditionalInfo**

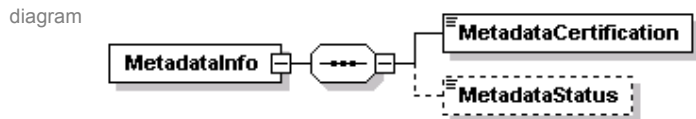


namespace urn:axmedis:01

children [MetadataInfo](#)

description contains additional information on metadata stored in other descriptors next to the AXInfo (DC, MPEG7, etc.)

element **AXInfo/MetadataAdditionalInfo/MetadataInfo**



namespace urn:axmedis:01

children [MetadataCertification](#) [MetadataStatus](#)

attributes	Name for	Type	Use	Default	Fixed
		xs:IDREF			

description contains additional info on the metadata descriptor stated by the for attribute

constraints the *for* attribute refers to a descriptor element next to the AXInfo or for the AXInfo

element **AXInfo/MetadataAdditionalInfo/MetadataInfo/MetadataCertification**

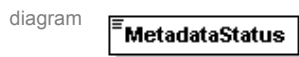


namespace urn:axmedis:01

type extension of **xs:string**

description contains certification information for a metadata descriptor

element **AXInfo/MetadataAdditionalInfo/MetadataInfo/MetadataStatus**



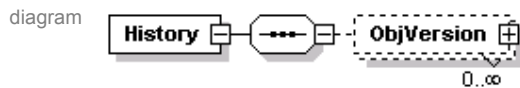
namespace urn:axmedis:01

DE3.1.2A – Framework and Tools Specification (General and Model)

type **xs:string**

description contains the editorial status of the metadata descriptor (e.g. to be completed, verified, ...)

element **History**

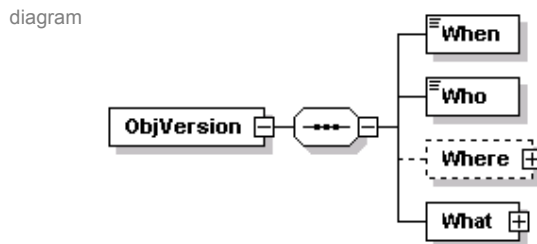


namespace urn:axmedis:01

children [ObjVersion](#)

description contains the history of the object

element **History/ObjVersion**



namespace urn:axmedis:01

children [When](#) [Who](#) [Where](#) [What](#)

attributes	Name	Type	Use	Default	Fixed
	number	xs:nonNegativeInteger			

description contains information on the history of a specific version of the object, the *number* attribute indicates the version number

element **History/ObjVersion/When**



namespace urn:axmedis:01

type **xs:dateTime**

description contains the date & time when the version was uploaded on the AXDB

element **History/ObjVersion/Who**

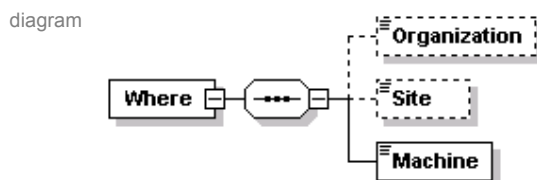


namespace urn:axmedis:01

type **xs:string**

description contains the name of the person who uploaded the object on the AXDB

element **History/ObjVersion/Where**



namespace urn:axmedis:01

DE3.1.2A – Framework and Tools Specification (General and Model)

children [Organization Site Machine](#)

description contains the indication of the location where the upload was performed

element History/ObjVersion/Where/Organization



namespace urn:axmedis:01

type **xs:string**

description contains the indication of the Organization where the upload was performed

element History/ObjVersion/Where/Site



namespace urn:axmedis:01

type **xs:string**

description contains the indication of the site where the upload was performed

element History/ObjVersion/Where/Machine

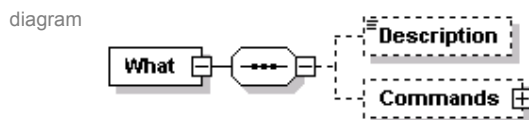


namespace urn:axmedis:01

type **xs:string**

description contains the indication of the machine where the upload was performed

element History/ObjVersion/What



namespace urn:axmedis:01

children [Description](#) [Commands](#)

description contains what have been performed on the object as a textual description and as the list of commands performed.

element History/ObjVersion/What/Description



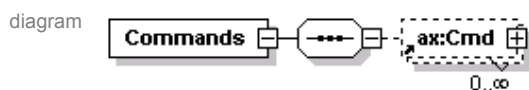
namespace urn:axmedis:01

type extension of **xs:string**

attributes	Name	Type	Use	Default	Fixed
	lang	xs:string	optional		

description contains textual description of what have been done on the object for the specific object version

element History/ObjVersion/What/Commands



namespace urn:axmedis:01

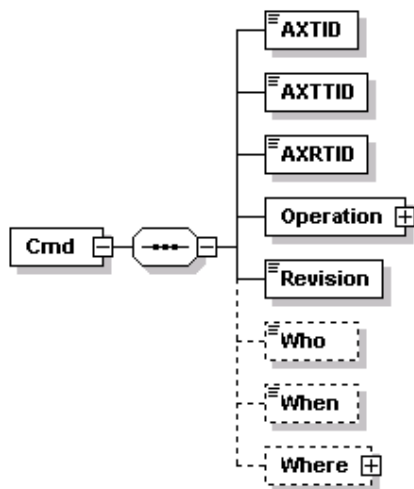
DE3.1.2A – Framework and Tools Specification (General and Model)

children [ax:Cmd](#)

description contains the commands performed on the object.

element **Cmd**

diagram



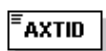
namespace urn:axmedis:01

children [AXTID](#) [AXTTID](#) [AXRTID](#) [Operation](#) [Revision](#) [Who](#) [When](#) [Where](#)

description contains information regarding a command performed on the object

element **Cmd/AXTID**

diagram



namespace urn:axmedis:01

type **xs:string**

description contains the AXMEDIS Tool ID identifying the tool used to perform the command

element **Cmd/AXTTID**

diagram



namespace urn:axmedis:01

type **xs:string**

description contains the AXMEDIS Tool Type ID identifying the type of tool used to perform the command

element **Cmd/AXRTID**

diagram



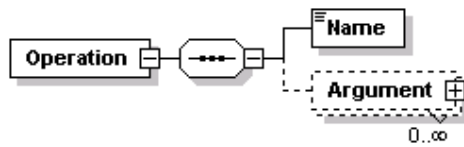
namespace urn:axmedis:01

type **xs:string**

description contains the AXMEDIS Real Tool ID identifying the tool instance used to produce the object

element **Cmd/Operation**

diagram



namespace urn:axmedis:01

children [Name](#) [Argument](#)

description contains the operation performed to the object

element **Cmd/Operation/Name**

diagram



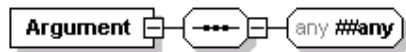
namespace urn:axmedis:01

type **xs:string**

description contains the name of the operation performed on the object

element **Cmd/Operation/Argument**

diagram

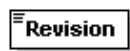


namespace urn:axmedis:01

description contains an argument for the operation, it can be any xml tag.

element **Cmd/Revision**

diagram



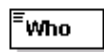
namespace urn:axmedis:01

type **xs:nonNegativeInteger**

description contains the revision number to which the command contributes

element **Cmd/Who**

diagram



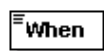
namespace urn:axmedis:01

type **xs:string**

description contains information regarding who performed the operation

element **Cmd/When**

diagram



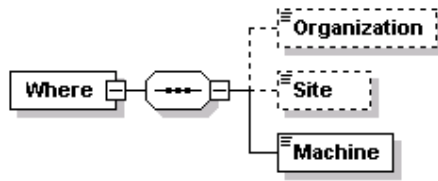
namespace urn:axmedis:01

type **xs:dateTime**

description contains when (date & time) the operation was performed

element Cmd/Where

diagram



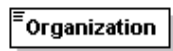
namespace urn:axmedis:01

children [Organization](#) [Site](#) [Machine](#)

description contains the location where the operation was performed

element Cmd/Where/Organization

diagram



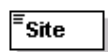
namespace urn:axmedis:01

type **xs:string**

description contains the Organization where the operation was performed

element Cmd/Where/Site

diagram



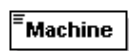
namespace urn:axmedis:01

type **xs:string**

description contains the site where the operation was performed

element Cmd/Where/Machine

diagram



namespace urn:axmedis:01

type **xs:string**

description contains the identifier of the machine where the operation was performed

The following is the complete textual description of the AXInfo Schema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:axmedis:01" xmlns:ax="urn:axmedis:01" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="urn:mpeg:mpeg21:2002:02-DIDL-NS" schemaLocation="xmlschemas/DIDL.xsd"/>
  <xs:import namespace="urn:mpeg:mpeg21:2002:01-DII-NS" schemaLocation="xmlschemas/dii.xsd"/>
  <xs:import namespace="urn:mpeg:mpeg21:2003:01-REL-R-NS" schemaLocation="xmlschemas/rel-r.xsd"/>
  <xs:import namespace="urn:mpeg:mpeg21:2003:01-REL-MX-NS" schemaLocation="xmlschemas/rel-mx.xsd"/>
  <xs:import namespace="urn:mpeg:mpeg21:2003:01-REL-SX-NS" schemaLocation="xmlschemas/rel-sx.xsd"/>
  <xs:element name="AXInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ObjectCreator" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="AXCID" type="xs:string"/>
              <xs:element name="ObjectCreatorName" type="xs:string" minOccurs="0"/>
              <xs:element name="ObjectCreatorURL" type="xs:anyURI" minOccurs="0"/>
              <xs:element name="ObjectCreatorCompany" type="xs:string" minOccurs="0"/>
              <xs:element name="ObjectCreatorCompanyURL" type="xs:anyURI" minOccurs="0"/>
              <xs:element name="ObjectCreatorNationality" type="xs:string"/>
            
```

```

</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Owner" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="OwnerID">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="coding" type="xs:string" use="required"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="OwnerName" type="xs:string" minOccurs="0"/>
      <xs:element name="OwnerURL" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="OwnerCompany" type="xs:string" minOccurs="0"/>
      <xs:element name="OwnerCompanyURL" type="xs:string" minOccurs="0"/>
      <xs:element name="OwnerNationality" type="xs:string"/>
      <xs:element name="OwnerDescription" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="lang" type="xs:string" use="required"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Distributor" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AXDID" type="xs:string"/>
      <xs:element name="DistributorName" type="xs:string" minOccurs="0"/>
      <xs:element name="DistributorURL" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="DistributorNationality" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="AccessMode">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="readOnly"/>
      <xs:enumeration value="read_write"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="CreationDate" type="xs:dateTime"/>
<xs:element name="LastModificationDate" type="xs:dateTime"/>
<xs:element name="Version" type="xs:nonNegativeInteger"/>
<xs:element name="Revision" type="xs:nonNegativeInteger"/>
<xs:element name="ObjectStatus" type="xs:string"/>
<xs:element name="ObjectType">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="BASIC"/>
      <xs:enumeration value="COMPOSITE"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="ObjectIsProtected" type="xs:boolean" minOccurs="0"/>
<xs:element name="ProtectionStamp" type="xs:string" minOccurs="0"/>
<xs:element name="ObjectIsGoverned" type="xs:boolean" minOccurs="0"/>
<xs:element name="IsPromoOf" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AXOID" type="xs:token" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

```

```

</xs:complexType>
</xs:element>
<xs:element ref="ax:History" minOccurs="0"/>
<xs:element name="Workflow" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="WorkItemID" type="xs:string"/>
      <xs:element name="WorkspaceInstanceID" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Fingerprints" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FingerprintID" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string"/>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="for" type="xs:IDREF" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="InternalPotentialAvailableRights" type="ax:PotentialAvailableRightsType" minOccurs="0"/>
<xs:element name="PotentialAvailableRights" type="ax:PotentialAvailableRightsType" minOccurs="0"/>
<xs:element name="MetadataAdditionalInfo" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="MetadataInfo" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="MetadataCertification">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string"/>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="MetadataStatus" type="xs:string" minOccurs="0"/>
          </xs:sequence>
          <xs:attribute name="for" type="xs:IDREF"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:element name="Cmd">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AXTID" type="xs:string"/>
      <xs:element name="AXTTID" type="xs:string"/>
      <xs:element name="AXRTID" type="xs:string"/>
      <xs:element name="Operation">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="Argument" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:any namespace="##any"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="Revision" type="xs:nonNegativeInteger"/>
<xs:element name="Who" type="xs:string" minOccurs="0"/>
<xs:element name="When" type="xs:dateTime" minOccurs="0"/>
<xs:element name="Where" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Organization" type="xs:string" minOccurs="0"/>
      <xs:element name="Site" type="xs:string" minOccurs="0"/>
      <xs:element name="Machine" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="History">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ObjVersion" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="When" type="xs:dateTime"/>
            <xs:element name="Who" type="xs:string"/>
            <xs:element name="Where" minOccurs="0">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Organization" type="xs:string" minOccurs="0"/>
                  <xs:element name="Site" type="xs:string" minOccurs="0"/>
                  <xs:element name="Machine" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="What">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Description" minOccurs="0">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:string">
                          <xs:attribute name="lang" type="xs:string" use="optional"/>
                        </xs:extension>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="Commands" minOccurs="0">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element ref="ax:Cmd" minOccurs="0" maxOccurs="unbounded"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="number" type="xs:nonNegativeInteger"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="PotentialAvailableRightsType">
  <xs:sequence>
    <xs:element name="LicensingURL" type="xs:anyURI"/>
    <xs:element name="PARStatus" type="xs:string" minOccurs="0"/>
    <xs:element ref="r:license"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

8.1.2.2 Dublin Core Metadata

The Dublin Core Metadata Initiative produced RDF schemas and XML schemas to allow the representation of Dublin Core metadata (for details see <http://dublincore.org/>) AXMEDIS will use this schemas to represent basic metadata.

The 15 basic metadata terms defined in Dublin Core are:

- contributor
- coverage
- creator
- date
- description
- format
- identifier
- language
- publisher
- relation
- rights
- source
- subject
- title
- type

each term may be repeated more than one time meaning that all of them applies to the resource described. Terms may be written in a different language and the language used is identified by a xml:lang attribute. A resource with:

```
<dc:creator>J. Doe<dc:creator>
<dc:creator>M. White<dc:creator>
<dc:title xml:lang="en">A title<dc:title>
<dc:title xml:lang="it">Un titolo<dc:title>
```

has two authors (J. Doe and M. White) and a title expressed in English and Italian.

In the following table is reported the definition for the DC terms as found in (<http://dublincore.org/documents/dcmi-terms/>).

contributor	
URI:	http://purl.org/dc/elements/1.1/contributor
Definition:	An entity responsible for making contributions to the content of the resource.
Comment:	Examples of a Contributor include a person, an organisation, or a service. Typically, the name of a Contributor should be used to indicate the entity.
coverage	
URI:	http://purl.org/dc/elements/1.1/coverage
Definition:	The extent or scope of the content of the resource.
Comment:	Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.
References:	[TGN] http://www.getty.edu/research/tools/vocabulary/tgn/index.html
creator	

DE3.1.2A – Framework and Tools Specification (General and Model)

URI:	http://purl.org/dc/elements/1.1/creator
Definition:	An entity primarily responsible for making the content of the resource.
Comment:	Examples of a Creator include a person, an organisation, or a service. Typically, the name of a Creator should be used to indicate the entity.
date	
URI:	http://purl.org/dc/elements/1.1/date
Definition:	A date associated with an event in the life cycle of the resource.
Comment:	Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and follows the YYYY-MM-DD format.
References:	[W3CDTF] http://www.w3.org/TR/NOTE-datetime
description	
URI:	http://purl.org/dc/elements/1.1/description
Definition:	An account of the content of the resource.
Comment:	Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.
format	
URI:	http://purl.org/dc/elements/1.1/format
Definition:	The physical or digital manifestation of the resource.
Comment:	Typically, Format may include the media-type or dimensions of the resource. Format may be used to determine the software, hardware or other equipment needed to display or operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types [MIME] defining computer media formats).
References:	[MIME] http://www.isi.edu/in-notes/iana/assignments/media-types/media-types
identifier	
URI:	http://purl.org/dc/elements/1.1/identifier
Definition:	An unambiguous reference to the resource within a given context.
Comment:	Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Example formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).
language	
URI:	http://purl.org/dc/elements/1.1/language
Definition:	A language of the intellectual content of the resource.
Comment:	Recommended best practice is to use RFC 3066 [RFC3066], which, in conjunction with ISO 639 [ISO639], defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English, "akk" for Akkadian, and "en-GB" for English used in the United Kingdom.
References:	[RFC3066] http://www.ietf.org/rfc/rfc3066.txt
References:	[ISO639] http://www.loc.gov/standards/iso639-2/
publisher	
URI:	http://purl.org/dc/elements/1.1/publisher
Definition:	An entity responsible for making the resource available
Comment:	Examples of a Publisher include a person, an organisation, or a service. Typically, the name of a Publisher should be used to indicate the entity.
relation	
URI:	http://purl.org/dc/elements/1.1/relation
Definition:	A reference to a related resource.
Comment:	Recommended best practice is to reference the resource by means of a string or number conforming to a formal

DE3.1.2A – Framework and Tools Specification (General and Model)

	identification system.
rights	
URI:	http://purl.org/dc/elements/1.1/rights
Definition:	Information about rights held in and over the resource.
Comment:	Typically, a Rights element will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource.
source	
URI:	http://purl.org/dc/elements/1.1/source
Definition:	A reference to a resource from which the present resource is derived.
Comment:	The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system.
subject	
URI:	http://purl.org/dc/elements/1.1/subject
Definition:	The topic of the content of the resource.
Comment:	Typically, a Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
title	
URI:	http://purl.org/dc/elements/1.1/title
Definition:	A name given to the resource.
Comment:	Typically, a Title will be a name by which the resource is formally known.
Multiplicity:	0..many (title in many languages)
type	
URI:	http://purl.org/dc/elements/1.1/type
Definition:	The nature or genre of the content of the resource.
Comment:	Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary [DCMITYPE]). To describe the physical or digital manifestation of the resource, use the Format element.
References:	[DCMITYPE] http://dublincore.org/documents/dcmi-type-vocabulary/

Many other terms has been introduced as refinements of these basic terms like:

- *abstract* as refinement of *description*
- *alternative* as refinement of *title*
- *conformsTo* as refinement of *relation*
- etc.

The full list is reported in the following and additional information can be found in (<http://dublincore.org/documents/dcmi-terms/>).

abstract	
Definition:	A summary of the content of the resource.
Refines:	http://purl.org/dc/elements/1.1/description
accessRights	
Definition:	Information about who can access the resource or an indication of its security status.
Comment:	Access Rights may include information regarding access or restrictions based on privacy, security or other regulations.
Refines:	http://purl.org/dc/elements/1.1/rights
alternative	

DE3.1.2A – Framework and Tools Specification (General and Model)

Definition:	Any form of the title used as a substitute or alternative to the formal title of the resource.
Comment:	This qualifier can include Title abbreviations as well as translations.
Refines:	http://purl.org/dc/elements/1.1/title
audience	
Definition:	A class of entity for whom the resource is intended or useful.
Comment:	A class of entity may be determined by the creator or the publisher or by a third party.
available	
Definition:	Date (often a range) that the resource will become or did become available.
Refines:	http://purl.org/dc/elements/1.1/date
bibliographicCitation	
Definition:	A bibliographic reference for the resource.
Comment:	Recommended practice is to include sufficient bibliographic detail to identify the resource as unambiguously as possible, whether or not the citation is in a standard form.
Refines:	http://purl.org/dc/elements/1.1/identifier
conformsTo	
Definition:	A reference to an established standard to which the resource conforms.
Refines:	http://purl.org/dc/elements/1.1/relation
created	
Definition:	Date of creation of the resource.
Refines:	http://purl.org/dc/elements/1.1/date
dateAccepted	
Definition:	Date of acceptance of the resource (e.g. of thesis by university department, of article by journal, etc.).
Refines:	http://purl.org/dc/elements/1.1/date
dateCopyrighted	
Definition:	Date of a statement of copyright.
Refines:	http://purl.org/dc/elements/1.1/date
dateSubmitted	
Definition:	Date of submission of the resource (e.g. thesis, articles, etc.).
Refines:	http://purl.org/dc/elements/1.1/date
educationLevel	
Definition:	A general statement describing the education or training context. Alternatively, a more specific statement of the location of the audience in terms of its progression through an education or training context.
Refines:	http://purl.org/dc/terms/audience
extent	
Definition:	The size or duration of the resource.
Refines:	http://purl.org/dc/elements/1.1/format
hasFormat	
Definition:	The described resource pre-existed the referenced resource, which is essentially the same intellectual content presented in another format.
Refines:	http://purl.org/dc/elements/1.1/relation
hasPart	
Definition:	The described resource includes the referenced resource either physically or logically.
Refines:	http://purl.org/dc/elements/1.1/relation

DE3.1.2A – Framework and Tools Specification (General and Model)

hasVersion	
Definition:	The described resource has a version, edition, or adaptation, namely, the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
isFormatOf	
Definition:	The described resource is the same intellectual content of the referenced resource, but presented in another format.
Refines:	http://purl.org/dc/elements/1.1/relation
isPartOf	
Definition:	The described resource is a physical or logical part of the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
isReferencedBy	
Definition:	The described resource is referenced, cited, or otherwise pointed to by the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
isReplacedBy	
Definition:	The described resource is supplanted, displaced, or superseded by the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
isRequiredBy	
Definition:	The described resource is required by the referenced resource, either physically or logically.
Refines:	http://purl.org/dc/elements/1.1/relation
issued	
Definition:	Date of formal issuance (e.g., publication) of the resource.
Refines:	http://purl.org/dc/elements/1.1/date
isVersionOf	
Definition:	The described resource is a version, edition, or adaptation of the referenced resource. Changes in version imply substantive changes in content rather than differences in format.
Refines:	http://purl.org/dc/elements/1.1/relation
license	
Definition:	A legal document giving official permission to do something with the resource.
Comment:	Recommended best practice is to identify the license using a URI. Examples of such licenses can be found at http://creativecommons.org/licenses/ .
Refines:	http://purl.org/dc/elements/1.1/rights
mediator	
Definition:	A class of entity that mediates access to the resource and for whom the resource is intended or useful.
Comment:	The audiences for a resource are of two basic classes: (1) an ultimate beneficiary of the resource, and (2) frequently, an entity that mediates access to the resource. The mediator element refinement represents the second of these two classes.
Refines:	http://purl.org/dc/terms/audience
medium	
Definition:	The material or physical carrier of the resource.
Refines:	http://purl.org/dc/elements/1.1/format
modified	
Definition:	Date on which the resource was changed.
Refines:	http://purl.org/dc/elements/1.1/date
provenance	
Definition:	A statement of any changes in ownership and custody of the resource since its creation that are significant for its authenticity, integrity and interpretation.

DE3.1.2A – Framework and Tools Specification (General and Model)

Comment:	The statement may include a description of any changes successive custodians made to the resource.
references	
Definition:	The described resource references, cites, or otherwise points to the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
replaces	
Definition:	The described resource supplants, displaces, or supersedes the referenced resource.
Refines:	http://purl.org/dc/elements/1.1/relation
requires	
Definition:	The described resource requires the referenced resource to support its function, delivery, or coherence of content.
Refines:	http://purl.org/dc/elements/1.1/relation
rightsHolder	
Definition:	A person or organization owning or managing rights over the resource.
Comment:	Recommended best practice is to use the URI or name of the Rights Holder to indicate the entity.
spatial	
Definition:	Spatial characteristics of the intellectual content of the resource.
Refines:	http://purl.org/dc/elements/1.1/coverage
tableOfContents	
Definition:	A list of subunits of the content of the resource.
Refines:	http://purl.org/dc/elements/1.1/description
temporal	
Definition:	Temporal characteristics of the intellectual content of the resource.
Refines:	http://purl.org/dc/elements/1.1/coverage
valid	
Definition:	Date (often a range) of validity of a resource.
Refines:	http://purl.org/dc/elements/1.1/date

AXMEDIS will support all these metadata (basic and refined), however in case of collision with information stored in other descriptors like in *AXInfo* or *Identifiers*, these ones are considered valid and the DC ones are dependent. Meaning that in case of inconsistency between these information the AXInfo and the Identifiers have a higher priority and can be used to fix the DC values (under user control).

Have to be noted that not all refined elements may have sense in the AXMEDIS context, thus some of them may be not considered by some applications (e.g. DB may not index some metadata).

8.1.3 Examples of AXMEDIS Objects

Basic AXMEDIS Object

The following is an example of a Basic AXMEDIS Object

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS" xmlns:ax="urn:axmedis:01" xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS" xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS" xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:axmedis:01 AXMEDIS.xsd">
  <Item>
    <!--
      Descriptor containing the AXOID to identify the object (REQUIRED)
    -->
    <Descriptor id="dsc_id">
      <Statement mimeType="text/xml">
```

```

    <dii:Identifier>urn:mpegRA:mpeg21:dii:axoid:A001AGSHDI</dii:Identifier>
  </Statement>
</Descriptor>
<Item>
  <!--
    Descriptor containing the RelatedIdentifier to identify the work (OPTIONAL)
  -->
  <Descriptor id="public_dsc_rel_id">
    <Statement mimeType="text/xml">
      <dii:RelatedIdentifier>urn:mpegRA:mpeg21:dii:iscw:T-034.524.680-1</dii:RelatedIdentifier>
    </Statement>
  </Descriptor>
  <!--
    Descriptor containing the AXInfo containing information regarding the object (REQUIRED)
  -->
  <Descriptor id="public_dsc_ax">
    <Statement mimeType="text/xml">
      <ax:AXInfo>
        <ax:ObjectCreator>
          <ax:AXCID>... a creator ID ...</ax:AXCID>
          <ax:ObjectCreatorName>J. Doe</ax:ObjectCreatorName>
          <ax:ObjectCreatorURL>mailto:jdoe@invideo.com</ax:ObjectCreatorURL>
          <ax:ObjectCreatorCompany>InVideo</ax:ObjectCreatorCompany>
          <ax:ObjectCreatorCompanyURL>http://www.invideo.com</ax:ObjectCreatorCompanyURL>
          <ax:ObjectCreatorNationality>US</ax:ObjectCreatorNationality>
        </ax:ObjectCreator>
        <ax:AccessMode>read_write</ax:AccessMode>
        <ax:CreationDate>2004-12-27T15:00:00</ax:CreationDate>
        <ax>LastModificationDate>2004-12-27T16:43:00</ax>LastModificationDate>
        <ax:Version>2</ax:Version>
        <ax:Revision>1</ax:Revision>
        <ax:ObjectStatus>production</ax:ObjectStatus>
        <ax:ObjectType>BASIC</ax:ObjectType>
      </ax:AXInfo>
      <!--
        History of the object
      -->
      <ax:History>
        <ax:ObjVersion number="1">
          <ax:When>2004-12-27T16:27:00</ax:When>
          <ax:Who>J. Doe</ax:Who>
          <ax:Where>
            <ax:Organization>InVideo</ax:Organization>
            <ax:Site>Atlanta</ax:Site>
            <ax:Machine>JDOE_01</ax:Machine>
          </ax:Where>
          <ax:What>
            <ax:Description>First version</ax:Description>
          </ax:What>
        </ax:ObjVersion>
        <ax:ObjVersion number="2">
          <ax:When>2004-12-27T16:27:00</ax:When>
          <ax:Who>J. Doe</ax:Who>
          <ax:Where>
            <ax:Organization>InVideo</ax:Organization>
            <ax:Site>Atlanta</ax:Site>
            <ax:Machine>JDOE_05</ax:Machine>
          </ax:Where>
          <ax:What>
            <ax:Commands>
              <ax:Cmd>
                <ax:AXTID>.... a Tool ID ...</ax:AXTID>
                <ax:AXTTID>... a Tool Type ID...</ax:AXTTID>
                <ax:AXRTID>... a Real Tool ID ...</ax:AXRTID>
                <ax:Operation>
                  <ax:Name>Add</ax:Name>
                </ax:Operation>
                <ax:Revision>1</ax:Revision>
              </ax:Cmd>
              <!-- To be completed -->
            </ax:Commands>
          </ax:What>
        </ax:ObjVersion>
      </ax:History>
    </ax:AXInfo>
  </Statement>
</Descriptor>
</Item>

```

```

</ax:ObjVersion>
</ax:History>
<!--
Workflow information
-->
<ax:Workflow>
  <ax:WorkItemID>... a work item ID... </ax:WorkItemID>
  <ax:WorkspaceInstanceID>.... a workspace instance ID ...</ax:WorkspaceInstanceID>
</ax:Workflow>
<!--
Fingerprint algorithms to be used for the resource
-->
<ax:Fingerprints for="cmp">
  <ax:FingerprintID>FngVideo01</ax:FingerprintID>
</ax:Fingerprints>
<!--
Rights potentially available on the object
-->
<ax:PotentialAvailableRights>
  <ax:LicensingURL>http://www.axmedis.org</ax:LicensingURL>
  <ax:PARStatus/>
  <r:license>
    <r:grantGroup>
      <r:grant>
        <mx:play/>
        <r:allConditions>
          <sx:validityIntervalFloating>
            <sx:duration>P1M</sx:duration>
          </sx:validityIntervalFloating>
          <r:validityInterval>
            <r:notAfter>2010-01-01T00:00:00</r:notAfter>
          </r:validityInterval>
        </r:allConditions>
      </r:grant>
      <r:grant>
        <mx:move/>
      </r:grant>
      <r:grant>
        <mx:delete/>
      </r:grant>
    </r:grantGroup>
  </r:license>
</ax:PotentialAvailableRights>
<!--
Information for metadata certification ...(to be better defined)
-->
<ax:MetadataAdditionalInfo>
  <ax:MetadataInfo for="dsc_id">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_rel_id">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_dc">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    <ax:MetadataStatus>partial</ax:MetadataStatus>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_mpeg7">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    <ax:MetadataStatus>partial</ax:MetadataStatus>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_ax">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
</ax:MetadataAdditionalInfo>
</ax:AXInfo>
</Statement>
</Descriptor>
<!--
Descriptor containing the Dublin Core information regarding the object (REQUIRED)
-->

```

```

<Descriptor id="public_dsc_dc">
  <Statement mimeType="text/xml">
    <rdf:Description>
      <dc:title xml:lang="en">When the Thistle Blooms</dc:title>
      <dc:title xml:lang="it">Quando il Cardo Sboccia</dc:title>
      <dc:creator>Always Red</dc:creator>
      <dc:publisher>PDQ Records</dc:publisher>
    </rdf:Description>
  </Statement>
</Descriptor>
<!--
  Descriptor containing the MPEG7 information regarding the object (OPTIONAL)
-->
<Descriptor id="public_dsc_mpeg7">
  <Statement mimeType="text/xml">
    <mpeg7:Mpeg7>
      <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
        <mpeg7:MediaFormat>
          <mpeg7:VisualCodingFormat
href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.2"/>
            <mpeg7:BitRate>64000</mpeg7:BitRate>
          </mpeg7:MediaFormat>
          <mpeg7:MediaQuality>
            <mpeg7:QualityRating ratingType="objective">
              <mpeg7:RatingValue>35.6</mpeg7:RatingValue>
              <mpeg7:RatingMetric>
                <mpeg7:QualityRatingScheme href="urn:mpeg:mpeg7:cs:MPEG-
7QualityRatingSchemeCS:2.3"/>
              <mpeg7:RatingStyle>higherBetter</mpeg7:RatingStyle>
            </mpeg7:RatingMetric>
          </mpeg7:QualityRating>
        </mpeg7:MediaQuality>
      </mpeg7:DescriptionUnit>
    </mpeg7:Mpeg7>
  </Statement>
</Descriptor>
<Descriptor>
  <Descriptor><Statement mimeType="text/plain">Reference to the real content</Statement></Descriptor>
  <Statement mimeType="text/uri-list">urn:axmedis:A001AGSHDI#A001AGSHDI_content</Statement>
</Descriptor>
</Item>
<Item id="A001AGSHDI_content">
  <!--
    Descriptor containing the RelatedIdentifier to identify the work (OPTIONAL)
  -->
  <Descriptor id="private_dsc_rel_id">
    <Statement mimeType="text/xml">
      <dii:RelatedIdentifier>urn:mpegRA:mpeg21:dii:iscw:T-034.524.680-1</dii:RelatedIdentifier>
    </Statement>
  </Descriptor>
  <!--
    Descriptor containing the AXInfo containing information regarding the object (REQUIRED)
  -->
  <Descriptor id="private_dsc_ax">
    <Statement mimeType="text/xml">
      <ax:AXInfo>
        <ax:ObjectCreator>
          <ax:AXCID>... a creator ID ...</ax:AXCID>
          <ax:ObjectCreatorName>J. Doe</ax:ObjectCreatorName>
          <ax:ObjectCreatorURL>mailto:jdoe@invideo.com</ax:ObjectCreatorURL>
          <ax:ObjectCreatorCompany>InVideo</ax:ObjectCreatorCompany>
          <ax:ObjectCreatorCompanyURL>http://www.invideo.com</ax:ObjectCreatorCompanyURL>
          <ax:ObjectCreatorNationality>US</ax:ObjectCreatorNationality>
        </ax:ObjectCreator>
        <ax:AccessMode>read_write</ax:AccessMode>
        <ax:CreationDate>2004-12-27T15:00:00</ax:CreationDate>
        <ax>LastModificationDate>2004-12-27T16:43:00</ax>LastModificationDate>
        <ax:Version>2</ax:Version>
        <ax:Revision>1</ax:Revision>
        <ax:ObjectStatus>production</ax:ObjectStatus>
        <ax:ObjectType>BASIC</ax:ObjectType>
      </ax:AXInfo>
    </Statement>
  </Descriptor>

```

```

<!--
History of the object
-->
<ax:History>
  <ax:ObjVersion number="1">
    <ax:When>2004-12-27T16:27:00</ax:When>
    <ax:Who>J. Doe</ax:Who>
    <ax:Where>
      <ax:Organization>InVideo</ax:Organization>
      <ax:Site>Atlanta</ax:Site>
      <ax:Machine>JDOE_01</ax:Machine>
    </ax:Where>
    <ax:What>
      <ax:Description>First version</ax:Description>
    </ax:What>
  </ax:ObjVersion>
  <ax:ObjVersion number="2">
    <ax:When>2004-12-27T16:27:00</ax:When>
    <ax:Who>J. Doe</ax:Who>
    <ax:Where>
      <ax:Organization>InVideo</ax:Organization>
      <ax:Site>Atlanta</ax:Site>
      <ax:Machine>JDOE_05</ax:Machine>
    </ax:Where>
    <ax:What>
      <ax:Commands>
        <ax:Cmd>
          <ax:AXTID>... a Tool ID ...</ax:AXTID>
          <ax:AXTTID>... a Tool Type ID...</ax:AXTTID>
          <ax:AXRTID>... a Real Tool ID ...</ax:AXRTID>
          <ax:Operation>
            <ax:Name>Add</ax:Name>
          </ax:Operation>
          <ax:Revision>1</ax:Revision>
        </ax:Cmd>
        <!-- To be completed -->
      </ax:Commands>
    </ax:What>
  </ax:ObjVersion>
</ax:History>
<!--
Workflow information
-->
<ax:Workflow>
  <ax:WorkItemID>... a work item ID...</ax:WorkItemID>
  <ax:WorkspaceInstanceID>... a workspace instance ID ...</ax:WorkspaceInstanceID>
</ax:Workflow>
<!--
Fingerprint algorithms to be used for the resource
-->
<ax:Fingerprints for="cmp">
  <ax:FingerprintID>FngVideo01</ax:FingerprintID>
</ax:Fingerprints>
<!--
Rights potentially available on the object
-->
<ax:PotentialAvailableRights>
  <ax:LicensingURL>http://www.axmedis.org</ax:LicensingURL>
  <ax:PARStatus/>
  <r:license>
    <r:grantGroup>
      <r:grant>
        <mx:play/>
        <r:allConditions>
          <sx:validityIntervalFloating>
            <sx:duration>P1M</sx:duration>
          </sx:validityIntervalFloating>
          <r:validityInterval>
            <r:notAfter>2010-01-01T00:00:00</r:notAfter>
          </r:validityInterval>
        </r:allConditions>
      </r:grant>
    </r:grantGroup>
  </r:license>
</ax:PotentialAvailableRights>

```

```

        </r:grant>
        <r:grant>
            <mx:move/>
        </r:grant>
        <r:grant>
            <mx:delete/>
        </r:grant>
    </r:grantGroup>
</r:license>
</ax:PotentialAvailableRights>
<!--
    Information for metadata certification ...(to be better defined)
-->
<ax:MetadataAdditionalInfo>
    <ax:MetadataInfo for="dsc_id">
        <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    </ax:MetadataInfo>
    <ax:MetadataInfo for="private_dsc_rel_id">
        <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    </ax:MetadataInfo>
    <ax:MetadataInfo for="private_dsc_dc">
        <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
        <ax:MetadataStatus>partial</ax:MetadataStatus>
    </ax:MetadataInfo>
    <ax:MetadataInfo for="private_dsc_mpeg7">
        <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
        <ax:MetadataStatus>partial</ax:MetadataStatus>
    </ax:MetadataInfo>
    <ax:MetadataInfo for="private_dsc_ax">
        <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    </ax:MetadataInfo>
</ax:MetadataAdditionalInfo>
</ax:AXInfo>
</Statement>
</Descriptor>
<!--
    Descriptor containing the Dublin Core information regarding the object (REQUIRED)
-->
<Descriptor id="private_dsc_dc">
    <Statement mimeType="text/xml">
        <rdf:Description>
            <dc:title xml:lang="en">When the Thistle Blooms</dc:title>
            <dc:title xml:lang="it">Quando il Cardo Sboccia</dc:title>
            <dc:creator>Always Red</dc:creator>
            <dc:publisher>PDQ Records</dc:publisher>
        </rdf:Description>
    </Statement>
</Descriptor>
<!--
    Descriptor containing the MPEG7 information regarding the object (OPTIONAL)
-->
<Descriptor id="private_dsc_mpeg7">
    <Statement mimeType="text/xml">
        <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
                <mpeg7:MediaFormat>
                    <mpeg7:VisualCodingFormat
href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.2"/>
                    <mpeg7:BitRate>64000</mpeg7:BitRate>
                </mpeg7:MediaFormat>
                <mpeg7:MediaQuality>
                    <mpeg7:QualityRating ratingType="objective">
                        <mpeg7:RatingValue>35.6</mpeg7:RatingValue>
                        <mpeg7:RatingMetric>
                            <mpeg7:QualityRatingScheme href="urn:mpeg:mpeg7:cs:MPEG-
7QualityRatingSchemeCS:2.3"/>
                        </mpeg7:RatingMetric>
                    </mpeg7:QualityRating>
                </mpeg7:MediaQuality>
            </mpeg7:DescriptionUnit>

```



```

        </mpeg7:Mpeg7>
    </Statement>
</Descriptor>
<!--
    Component elements containing the resource (REQUIRED for single object)
-->
    <Component id="cmp">
        <Resource mimeType="video/mp4v-es" encoding="base64">
            aadsfadsfsyd647dgd78r85hfuv8nbr8fnf985nf9g9gm569gnty9ghmg90hdhd8fhfd9d9
            dh8f95mnfk9gfm59fgt95mkt0jhdf8fnj587fd67n3jf84mf00eedjf8fj58tm58fm58emds9o
            ...
        </Resource>
    </Component>
</Item>
</Item>
</DIDL>

```

Protected Basic AXMEDIS Object

```

<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS" xmlns:ipmpdidl="urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS"
xmlns:ipmp="urn:mpeg:mpeg21:2004:01-IPMP-NS" xmlns:ax="urn:axmedis:01" xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS" xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:axmedis:01 AXMEDIS.xsd">
    <Item>
        <!--
            Descriptor containing the AXOID to identify the object (REQUIRED)
        -->
        <Descriptor id="dsc_id">
            <Statement mimeType="text/xml">
                <dii:Identifier>urn:mpegRA:mpeg21:dii:axoid:A001AGSHDI</dii:Identifier>
            </Statement>
        </Descriptor>
    </Item>
        <!--
            Descriptor containing the RelatedIdentifier to identify the work (OPTIONAL)
        -->
        <Descriptor id="public_dsc_rel_id">
            <Statement mimeType="text/xml">
                <dii:RelatedIdentifier>urn:mpegRA:mpeg21:dii:iscw:T-034.524.680-1</dii:RelatedIdentifier>
            </Statement>
        </Descriptor>
        <!--
            Descriptor containing the AXInfo containing information regarding the object (REQUIRED)
        -->
        <Descriptor id="public_dsc_ax">
            <Statement mimeType="text/xml">
                <ax:AXInfo>
                    <ax:ObjectCreator>
                        <ax:AXCID>... a creator ID ...</ax:AXCID>
                        <ax:ObjectCreatorName>J. Doe</ax:ObjectCreatorName>
                        <ax:ObjectCreatorURL>mailto:jdoe@invideo.com</ax:ObjectCreatorURL>
                        <ax:ObjectCreatorCompany>InVideo</ax:ObjectCreatorCompany>
                        <ax:ObjectCreatorCompanyURL>http://www.invideo.com</ax:ObjectCreatorCompanyURL>
                        <ax:ObjectCreatorNationality>US</ax:ObjectCreatorNationality>
                    </ax:ObjectCreator>
                    <ax:AccessMode>read_write</ax:AccessMode>
                    <ax:CreationDate>2004-12-27T15:00:00</ax:CreationDate>
                    <ax:LastModificationDate>2004-12-27T16:43:00</ax:LastModificationDate>
                    <ax:Version>2</ax:Version>
                    <ax:Revision>1</ax:Revision>
                    <ax:ObjectStatus>production</ax:ObjectStatus>
                    <ax:ObjectType>BASIC</ax:ObjectType>
                </ax:AXInfo>
                <!--
                    History of the object
                -->
                <ax:History>
                    <ax:ObjVersion number="1">

```

```

<ax:When>2004-12-27T16:27:00</ax:When>
<ax:Who>J. Doe</ax:Who>
<ax:Where>
  <ax:Organization>InVideo</ax:Organization>
  <ax:Site>Atlanta</ax:Site>
  <ax:Machine>JDOE_01</ax:Machine>
</ax:Where>
<ax:What>
  <ax:Description>First version</ax:Description>
</ax:What>
</ax:ObjVersion>
<ax:ObjVersion number="2">
  <ax:When>2004-12-27T16:27:00</ax:When>
  <ax:Who>J. Doe</ax:Who>
  <ax:Where>
    <ax:Organization>InVideo</ax:Organization>
    <ax:Site>Atlanta</ax:Site>
    <ax:Machine>JDOE_05</ax:Machine>
  </ax:Where>
  <ax:What>
    <ax:Commands>
      <ax:Cmd>
        <ax:AXTID>.... a Tool ID ...</ax:AXTID>
        <ax:AXTTID>... a Tool Type ID...</ax:AXTTID>
        <ax:AXRTID>... a Real Tool ID ...</ax:AXRTID>
        <ax:Operation>
          <ax:Name>Add</ax:Name>
        </ax:Operation>
        <ax:Revision>1</ax:Revision>
      </ax:Cmd>
      <!-- To be completed -->
    </ax:Commands>
  </ax:What>
</ax:ObjVersion>
</ax:History>
<!--
  Workflow information
-->
<ax:Workflow>
  <ax:WorkItemID>... a work item ID...</ax:WorkItemID>
  <ax:WorkspaceInstanceID>.... a workspace instance ID ...</ax:WorkspaceInstanceID>
</ax:Workflow>
<!--
  Fingerprint algorithms to be used for the resource
-->
<ax:Fingerprints for="cmp">
  <ax:FingerprintID>FngVideo01</ax:FingerprintID>
</ax:Fingerprints>
<!--
  Rights potentially available on the object
-->
<ax:PotentialAvailableRights>
  <ax:LicensingURL>http://www.axmedis.org</ax:LicensingURL>
  <ax:PARStatus/>
  <r:license>
    <r:grantGroup>
      <r:grant>
        <mx:play/>
        <r:allConditions>
          <sx:validityIntervalFloating>
            <sx:duration>P1M</sx:duration>
          </sx:validityIntervalFloating>
          <r:validityInterval>
            <r:notAfter>2010-01-01T00:00:00</r:notAfter>
          </r:validityInterval>
        </r:allConditions>
      </r:grant>
      <r:grant>
        <mx:move/>
      </r:grant>
    </r:grantGroup>
  </r:license>

```

```

        <mx:delete/>
      </r:grant>
    </r:grantGroup>
  </r:license>
</ax:PotentialAvailableRights>
<!--
  Information for metadata certification ...(to be better defined)
-->
<ax:MetadataAdditionalInfo>
  <ax:MetadataInfo for="dsc_id">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_rel_id">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_dc">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    <ax:MetadataStatus>partial</ax:MetadataStatus>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_mpeg7">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
    <ax:MetadataStatus>partial</ax:MetadataStatus>
  </ax:MetadataInfo>
  <ax:MetadataInfo for="public_dsc_ax">
    <ax:MetadataCertification>... certification data ...</ax:MetadataCertification>
  </ax:MetadataInfo>
</ax:MetadataAdditionalInfo>
</ax:AXInfo>
</Statement>
</Descriptor>
<!--
  Descriptor containing the Dublin Core information regarding the object (REQUIRED)
-->
<Descriptor id="public_dsc_dc">
  <Statement mimeType="text/xml">
    <rdf:Description>
      <dc:title xml:lang="en">When the Thistle Blooms</dc:title>
      <dc:title xml:lang="it">Quando il Cardo Sboccia</dc:title>
      <dc:creator>Always Red</dc:creator>
      <dc:publisher>PDQ Records</dc:publisher>
    </rdf:Description>
  </Statement>
</Descriptor>
<!--
  Descriptor containing the MPEG7 information regarding the object (OPTIONAL)
-->
<Descriptor id="public_dsc_mpeg7">
  <Statement mimeType="text/xml">
    <mpeg7:Mpeg7>
      <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
        <mpeg7:MediaFormat>
          <mpeg7:VisualCodingFormat
href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.2"/>
          <mpeg7:BitRate>64000</mpeg7:BitRate>
        </mpeg7:MediaFormat>
        <mpeg7:MediaQuality>
          <mpeg7:QualityRating ratingType="objective">
            <mpeg7:RatingValue>35.6</mpeg7:RatingValue>
            <mpeg7:RatingMetric>
              <mpeg7:QualityRatingScheme href="urn:mpeg:mpeg7:cs:MPEG-
7QualityRatingSchemeCS:2.3"/>
              <mpeg7:RatingStyle>higherBetter</mpeg7:RatingStyle>
            </mpeg7:RatingMetric>
          </mpeg7:QualityRating>
        </mpeg7:MediaQuality>
      </mpeg7:DescriptionUnit>
    </mpeg7:Mpeg7>
  </Statement>
</Descriptor>
</Descriptor>
</Descriptor>

```

DE3.1.2A – Framework and Tools Specification (General and Model)

```

    <Statement mimeType="text/plain">Reference to the real content</Statement>
  </Descriptor>
  <Statement mimeType="text/uri-list">urn:axmedis:A001AGSHDI#A001AGSHDI_content</Statement>
</Descriptor>
</Item>
<ipmpdidl:Item>
  <ipmpdidl:Identifier>urn:axmedis:A001AGSHDI#A001AGSHDI_content</ipmpdidl:Identifier>
  <ipmpdidl:Info>
    <ipmp:IPMPInfoDescriptor>
      <ipmp:Tool>
        <ipmp:ToolBaseDescription>
          <ipmp:IPMPToolID>urn:mpegRA:mpeg21:IPMP:ABC005:77:29</ipmp:IPMPToolID>
          <ipmp:Remote ref="urn:IPMPToolsServer:ToolEnc005-3484"/>
        </ipmp:ToolBaseDescription>
      </ipmp:Tool>
      <ipmp:Tool>
        <ipmp:ToolBaseDescription>
          <ipmp:IPMPToolID>urn:mpegRA:mpeg21:IPMP:ABC064:55:86</ipmp:IPMPToolID>
          <ipmp:Remote ref="urn:IPMPToolsServer:ToolWat005-6393"/>
        </ipmp:ToolBaseDescription>
      </ipmp:Tool>
      <ipmp:RightsDescriptor>
        <ipmp:License>
          <r:license>
            <r:encryptedLicense Type="http://www.w3.org/2001/04/xmlenc#Content">
              <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#3des-cbc"/>
              <dsig:KeyInfo>
                <dsig:KeyName>SymmetricKey</dsig:KeyName>
              </dsig:KeyInfo>
              <enc:CipherData>
                <enc:CipherValue>Ktd63SDfkDWEjeSdkj39872A5ToQ...</enc:CipherValue>
              </enc:CipherData>
            </r:encryptedLicense>
          </r:license>
        </ipmp:License>
      </ipmp:RightsDescriptor>
    </ipmp:IPMPInfoDescriptor>
  </ipmpdidl:Info>
  <ipmpdidl:Contents>agsdhsjdddjfhf945734md9v784nf.... 7283udfhjdf94jdbnhcysd8e</ipmpdidl:Contents>
</ipmpdidl:Item>
</Item>
</DIDL>

```

8.2 AXMEDIS Model (DSI)

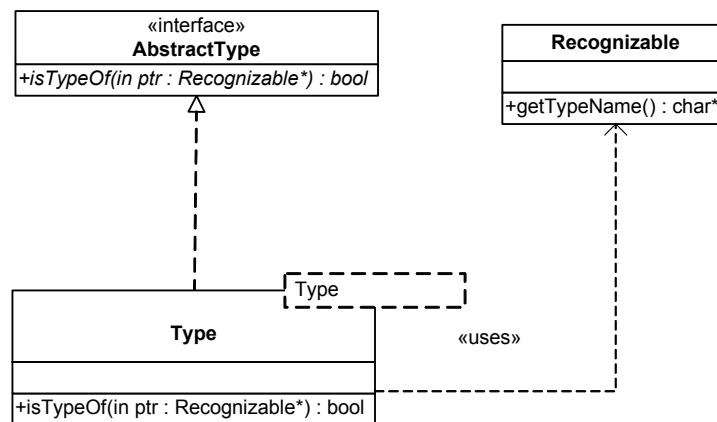
Module Profile		
AXMEDIS Model		
Executable or Library(Support)	Support library	
Single Thread or Multithread		
Language of Development	C++	
Responsible Name	Bellini, Vallotti, Rogai	
Responsible Partner	DSI	
Status (proposed/approved)	proposed	
Platforms supported		
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section

DE3.1.2A – Framework and Tools Specification (General and Model)

User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

8.2.1 MPEG21 Digital Item Model

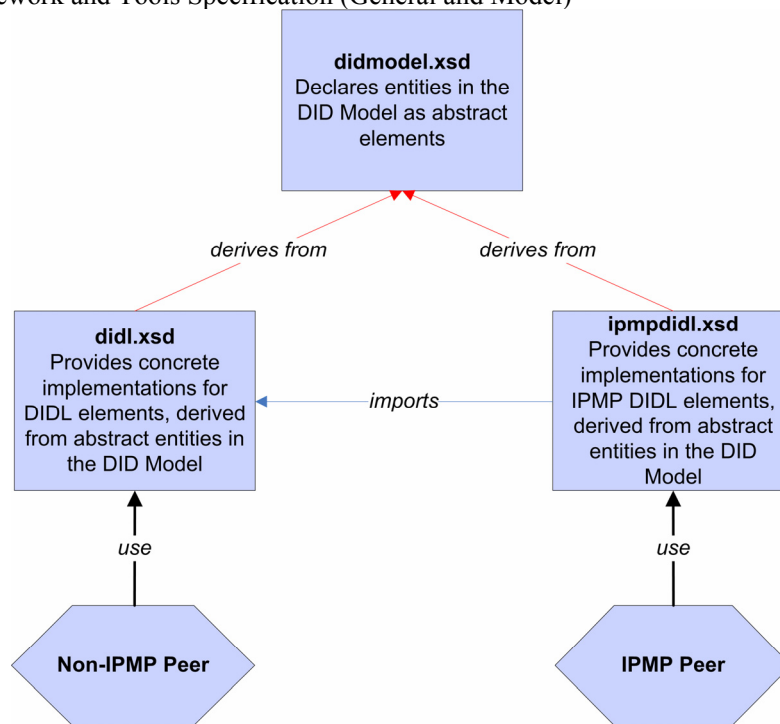
In this section classes modeling the MPEG21 digital items are reported.



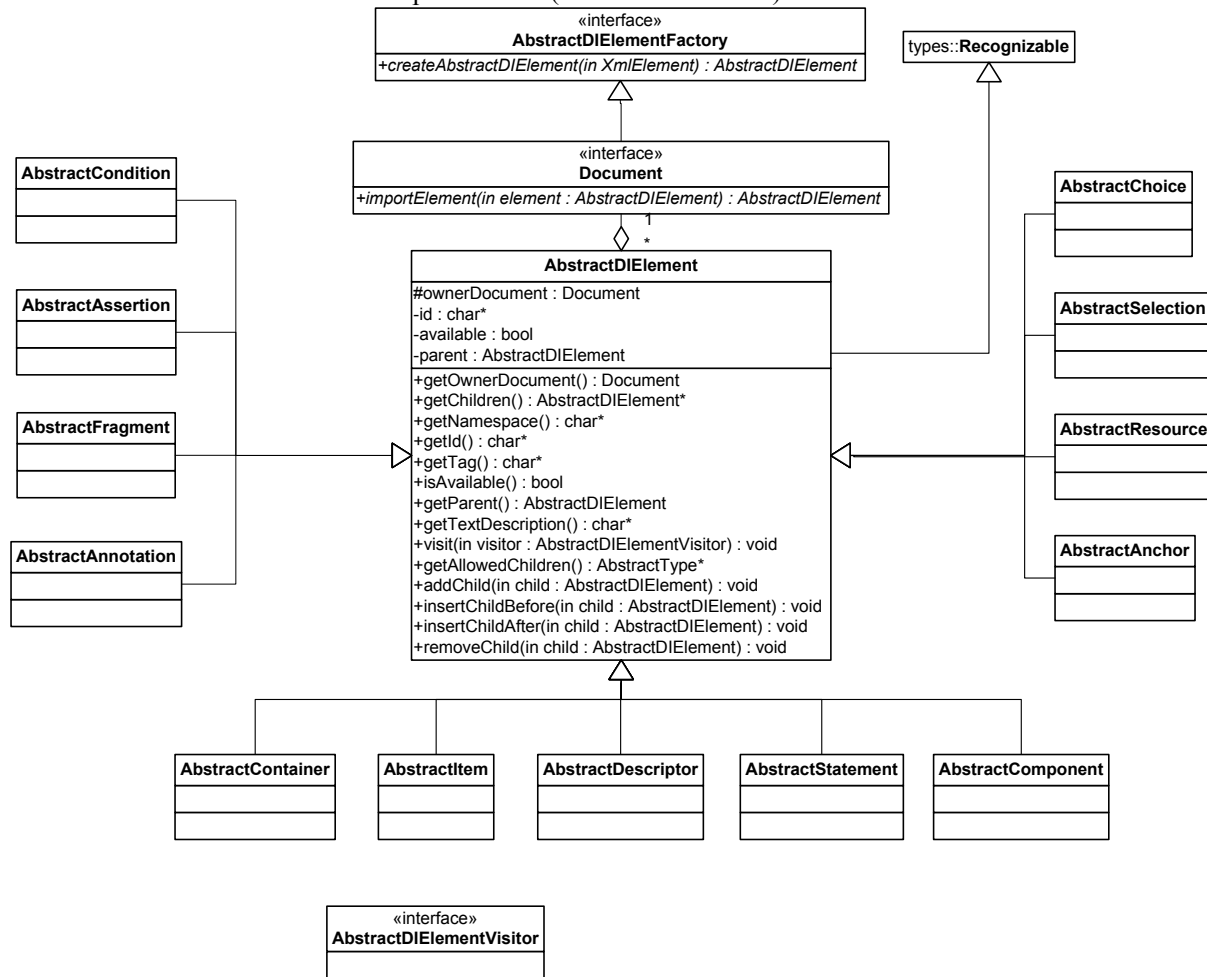
Since the development language is C++ a Type description class is needed in order to compare types of instances. A compare is necessary to control the proper insertion of elements in the data model. For example to enforce that the allowed children of a specific element are only a subset of the all elements at disposal, a list of Types is given. This list can be used to check if the insertion of a candidate element in the children is correct.

Any hierarchy of classes that want to support this feature have to be inherited from **Recognizable**. The class **AbstractType** is used to manage type information without taking care of the used actualization of the template class Type. Thus a reference to an AbstractType object can be passed as parameter, stored in an array and at any time it can be used to check if a given instance belongs to its hierarchy (if the instance has been constructed as a sub class of that described from the AbstractType reference).

Since AXMEDIS Objects are MPEG21 compliant objects the MPEG21 data model have to be managed. The following static structure diagrams take into account the MPEG21 Digital Item Description model (didmodel).



The model has been conceived to store a hierarchical structure of a multimedia object, each part of it can be protected or not. For this reason two different schema are provided one for the clear-text elements and the other for the protected ones. Both schema have derives from the same hierarchy specified in the didmodel which deals with abstract element

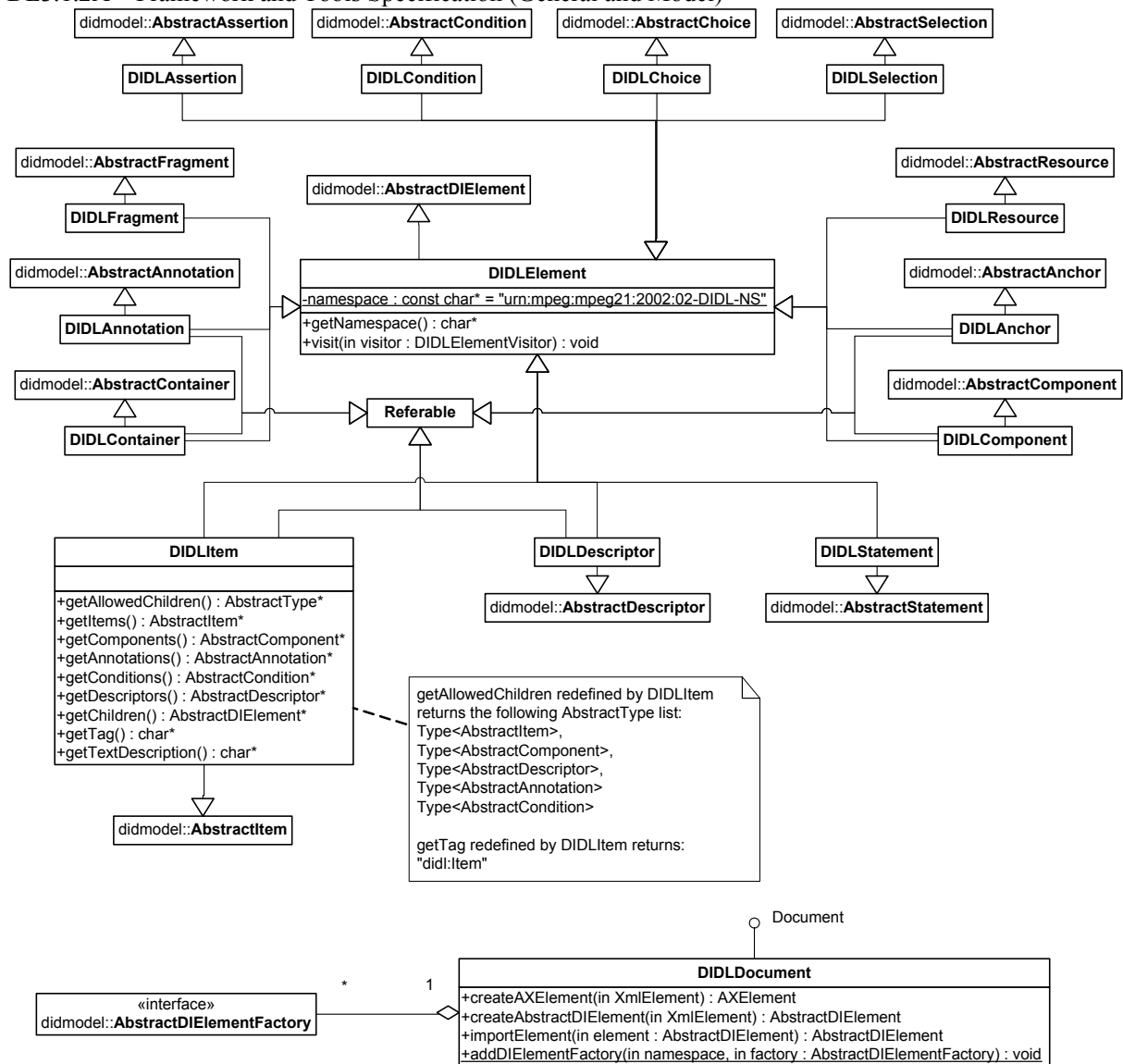


In the above diagram the classes that manage the abstract structure of the didmodel are shown. The base class **AbstractDIElement** is inherited from Recognizable class in order to support type recognition of all the subclasses. AbstractDIElement exposes common methods of all the DI elements i.e reference to the owner document, list of children, tag, id, structure management methods... many of the exposed methods are polymorphic and should be redefined from the sub-classes (i.e. *getAllowedChildren*). The abstract elements are the base to treat equally equivalent tag from protected and clear-text schema (e.g.. AbstractItem represent clear-text DIDLItem, protected IPMPItem).

AbstractDIElementFactory is the interface that could be implemented by any hierarchy (DIDL, IPMP) to allow automatic creation of elements.

Document interface should be implemented by those classes which want to register itself as the owner document of a hierarchy of elements (hierarchy manager).

DE3.1.2A – Framework and Tools Specification (General and Model)

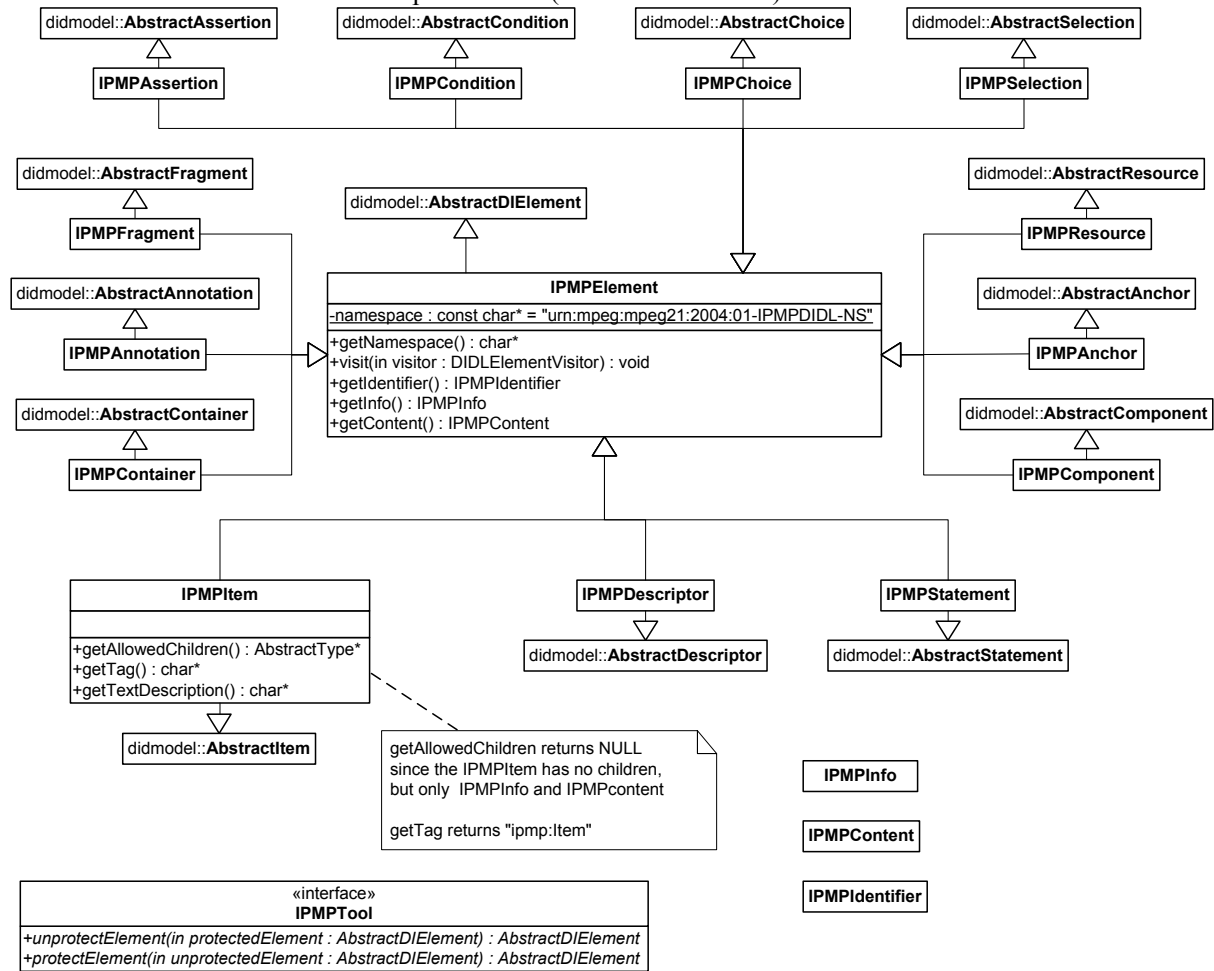


The above diagram maps all the elements which are treated as clear-text. It should be noted that each class representing such elements are inherited from the correspondent abstract. The **DIDLElement** class is another generalization of the whole set of elements since it overrides the method introduced from the AbstractDIElement that are in common among all the DIDL elements.

The note describe and example of how any of these element can override the remaining virtual methods like *getAllowedChildren*: this override will return a list of AbstractType instances.

Please note that **Referable** class is the generalization about elements that can be obtained with a reference, this class include the mechanisms to get the reference from the schema and to resolve that reference filling the element attributes.

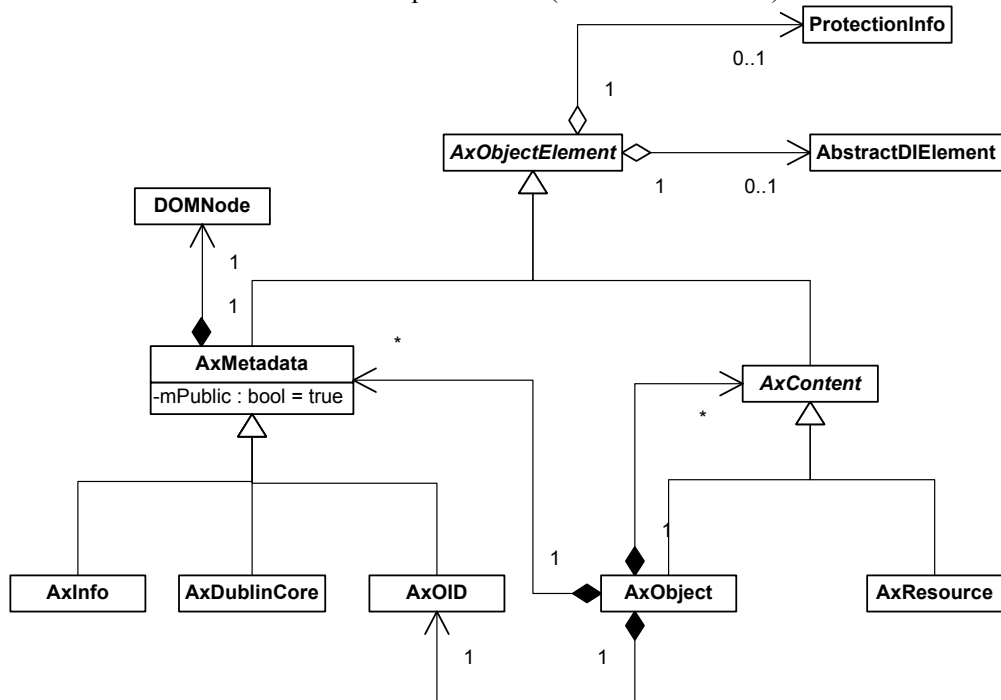
DE3.1.2A – Framework and Tools Specification (General and Model)



The above diagram describes the protected element hierarchy. The base class **IPMPElement** redefines the virtual methods differently from **DIDLElement**. It introduces the common methods to retrieve identifier, protection info, and protected content from all the IPMP elements (since they maintain the same structure).

8.2.2 AXMEDIS Object Model

In this section the classes modelling AXMEDIS objects are reported.

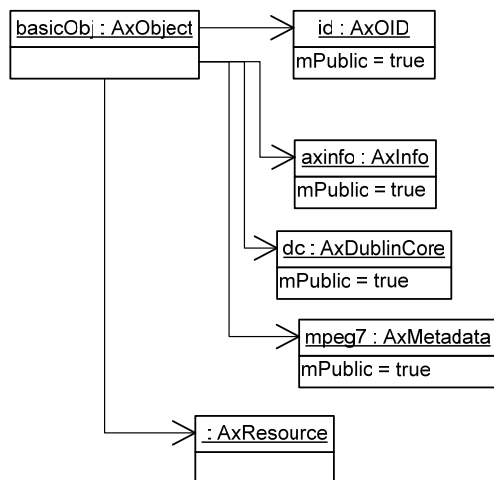


Abstract class *AxObjectElement* represents any element that can be stored in a AXMEDIS Object. It refers to an *AbstractDIElement* that represent it in a MPEG21 Digital Item and it may also refer to *ProtectionInfo* used to protect it.

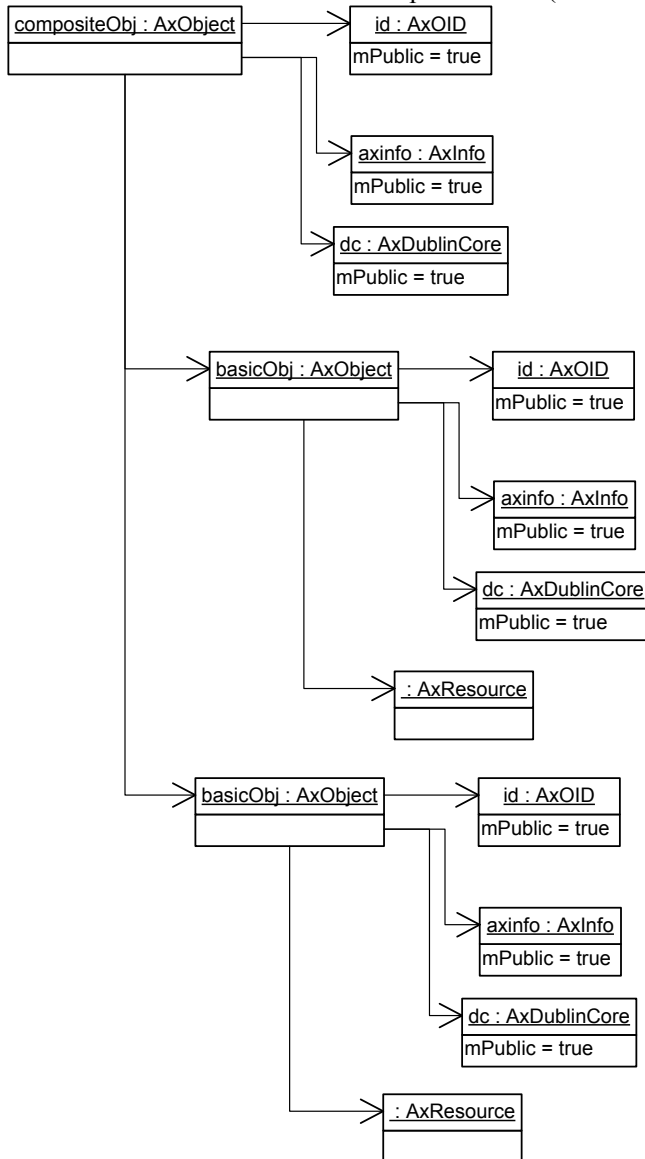
Class *AxMetadata* represents any XML metadata associated with a content, it is further specialised in *AxInfo*, *AxDublinCore* and *AxOID*. The `mPublic` attribute indicates if the metadata has to visible even if the object is protected.

Abstract class *AxContent* represents content to be stored in AXMEDIS objects, it can be *AxObject* or *AxResource*. An *AxResource* represents any digital resource identified with a mime type, it can be an image, a document, an audio. An *AxObject* can contain any number of metadata and any number of content.

The following is the object diagram of a basic object:



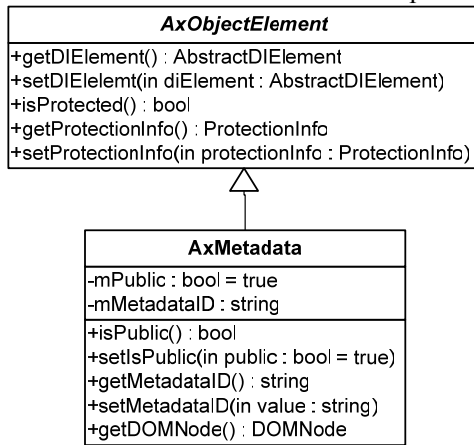
While the following is the object diagram of a composite AXMEDIS object:



8.2.2.1 AxMetadata

Class *AxMetadata* is a class to store any XML metadata.

The *AbstractDIElement* it refers to should be a *DIDLDescriptor*, containing a *DIDLStatement* with the XML content. XML content can be accessed through a *DOMNode* object.



8.2.2.2 AxInfo

Class *AxInfo*, derived from *AxMetadata*, provides access to the information related to the AXMEDIS object. Methods available for this class are:

ObjectCreator Management

+addObjectCreator(in position : int = -1) : int
adds a new ObjectCreator in the position given (starting from 0), position -1 means to add at the end. the return value indicates the position in which it is added.
+removeObjectCreator(in position : int)
removes an ObjectCreator from the position specified
+getObjectCreatorCount() : int
returns the number of ObjectCreator present
+findObjectCreatorByAXCID(in axcid : string) : int
returns the position of an ObjectCreator with a specific AXCID. It returns -1 if not found.
+findObjectCreatorByName(in name : string) : int
returns the position of an ObjectCreator with a specific Name. It returns -1 if not found.
+getObjectCreatorAXCID(in refNum : int = 0) : string
+setObjectCreatorAXCID(in refNum : int, in value : string)
allow to get and set the AXCID value for an ObjectCreator identified by position
+getObjectCreatorName(in refNum : int = 0) : string
+setObjectCreatorName(in refNum : int, in value : string)
allow to get and set the Name value for an ObjectCreator identified by position
+getObjectCreatorURL(in refNum : int = 0) : string
+setObjectCreatorURL(in refNum : int, in value : string)
allow to get and set the URL value for an ObjectCreator identified by position
+getObjectCreatorCompany(in refNum : int = 0) : string
+setObjectCreatorCompany(in refNum : int, in value : string)
allow to get and set the Company value for an ObjectCreator identified by position
+getObjectCreatorCompanyURL(in refNum : int = 0) : string
+setObjectCreatorCompanyURL(in refNum : int, in value : string)
allow to get and set the CompanyURL value for an ObjectCreator identified by position
+getObjectCreatorNationality(in refNum : int = 0) : string
+setObjectCreatorNationality(in refNum : int, in value : string)
allow to get and set the Nationality value for an ObjectCreator identified by position

Owner Management

+getOwnerID() : string
+setOwnerID(in value: string)
allow to get and set the code identifying the owner
+getOwnerIDCoding() : string

DE3.1.2A – Framework and Tools Specification (General and Model)

+setOwnerIDCoding(in value: string)
allow to get and set the coding used to identify the owner
+getOwnerName() : string
+setOwnerName(in value: string)
allow to get and set the name of the owner
+getOwnerURL() : string
+setOwnerURL(in value: string)
allow to get and set the URL of the owner
+getOwnerCompany() : string
+setOwnerCompany(in value: string)
allow to get and set the company of the owner
+getOwnerCompanyURL() : string
+setOwnerCompanyURL(in value: string)
allow to get and set the company URL of the owner
+getOwnerNationality() : string
+setOwnerNationality(in value: string)
allow to get and set the nationality of the owner
+addOwnerDescription(in position:int = -1) : int
adds a new description of the owner at the position specified or at the end if position is -1. The return value indicates the position where it is added.
+removeOwnerDescription(in position:int)
removes the description specified
+getOwnerDescription(in position:int = 0) : string
+setOwnerDescription(in position:int, in value:string)
allow to get and set the value of the description
+getOwnerDescriptionLanguage(in position:int = 0) : string
+setOwnerDescriptionLanguage(in position:int, in value:string)
allow to get and set the value of the description language

Distributor Management

+addDistributor()
adds a Distributor if not present.
+removeDistributor()
removes the Distributor
+getDistributorCount() : int
returns the number of Distributors present
+getDistributorAXDID() : string
+setDistributorAXDID(in value : string)
allow to get and set the AXDID value for the Distributor
+getDistributorName() : string
+setDistributorName(in value : string)
allow to get and set the Name value for the Distributor
+getDistributorURL() : string
+setDistributorURL(in value : string)
allow to get and set the URL value for the Distributor
+getDistributorNationality() : string
+setDistributorNationality(in value : string)
allow to get and set the Nationality value for the Distributor

Object Status

+getAccessMode() : string
+setAccessMode(in value : string)
allow to get and set the Access the the object “READ ONLY” or “READ WRITE”
+getCreationDate() : DateTime
get the date and time of object creation

DE3.1.2A – Framework and Tools Specification (General and Model)

+getLastModificationDate() : DateTime
get the date and time of object modification
+getVersion() : int
get the version of the object
+getRevision() : int
get the revision of the object
+getObjectStatus() : string
+setObjectStatus(in value : string)
allow to get and set the current status of the object
+getObjectType() : string
allow to get object type (“BASIC” or “COMPOSITE”)
+getObjectsProtected() : bool
+setObjectsProtected(in value:bool)
allows to get and set if the object is protected or not
+getProtectionStamp() : string
+setProtectionStamp(in value:string)
allows to get and set the protection stamp
+getObjectsGoverned() : bool
+setObjectsGoverned(in value:bool)
allow to get and set if the object contains a licence or not

PromoOf Management

+addPromoOfAXOID(in axoid:string, in position:int=-1)
adds a new AXOID in the PromoOf section, the position indicates where to put the AXOID, -1 means at the end
+removePromoOfAXOID(in position:int)
removes the AXOID in the position specified
+getPromoOfAXOIDCount() : int
get the count of AXOID in the PromoOf section
+getPromoOfAXOID(in position:int) : string
+setPromoOfAXOID(in position:int, in value:string)
allow to get and set the AXOID in a specified position

Workflow Status

+getWorkflowWorkItemID() : string
+setWorkflowWorkItemID (in value : string)
allow to get and set the WorkflowWorkItemID
+getWorkflowWorkspaceInstanceID() : string
+setWorkflowWorkspaceInstanceID (in value : string)
allow to get and set the WorkflowWorkspaceInstanceID

Fingerprints Management

+addFingerprints(in for:string, in position:int=-1) : int
adds a new Fingerprints section for a specified resource
+removeFingerprints(in position:int)
removes the Fingerprints section specified
+getFingerprintsCount() : int
gets the number of Fingerprints sections
+getFingerprintsFor(in position:int) : string
+setFingerprintsFor(in position:int, in value:string)
get and set the “for” attribute in the Fingerprints sections
+findFingerprintsFor(in for:string) : int
finds the fingerprint section for the specified resource, returns -1 if not found
+addFingerprintID(in fingerprints:int, in fingerprintId:string, in position:int=-1) : int
adds a new fingerprint ID in the Fingerprints section specified, the position argument indicates where to put the value (-1 is at the end)

DE3.1.2A – Framework and Tools Specification (General and Model)

+removeFingerprintID(in fingerprints:int, in position:int)
removes the fingerprinted in the Fingerprints section specified
+getFingerprintIDCount(in fingerprints:int)
gets the count of FingerprintID in the Fingerprints section specified
+getFingerprintID(in fingerprints:int, in position:int) : string
+setFingerprintID(in fingerprints:int, in position:int, in fingerprintid:string)
allow to get and set the fingerprintid

Internal Potential Available Rights Management

+addInternalPotentialAvailableRights()
adds a new Internal PAR section
+removeInternalPotentialAvailableRights()
removes the Internal PAR section
+getInternalPotentialAvailableRightsCount()
gets how many Internal PAR sections are present (0 or 1)
+getInternalPotentialAvailableRightsStatus() : string
+setInternalPotentialAvailableRightsStatus(in value:string)
allow to get and set the internal PAR status
+getInternalPotentialAvailableRightsLicense() : DOMNode
gets the DOM node of the license

Potential Available Rights Management

+addPotentialAvailableRights()
adds a new PAR section if not present
+removePotentialAvailableRights()
removes the PAR section
+getPotentialAvailableRightsCount()
gets how many PAR sections are present (0 or 1)
+getPotentialAvailableRightsLicensingURL() : string
+setPotentialAvailableRightsLicensingURL (in value:string)
allow to get and set the licensing URL
+getPotentialAvailableRightsStatus() : string
+setPotentialAvailableRightsStatus(in value:string)
allow to get and set the PAR status
+getPotentialAvailableRightsLicense() : DOMNode
gets the DOM node of the license

Additional Metadata Management

+getMetadataCertification(in for:string) : string
+setMetadataCertification(in for:string, in value:string)
allow to get and set the metadata certification for a specific descriptor
+getMetadataStatus(in for:string) : string
+setMetadataStatus(in for:string, in value:string)
allow to get and set the metadata status for a specific descriptor
+removeMetadataInfo(in for:string)
removes all the metadata info for a specific descriptor

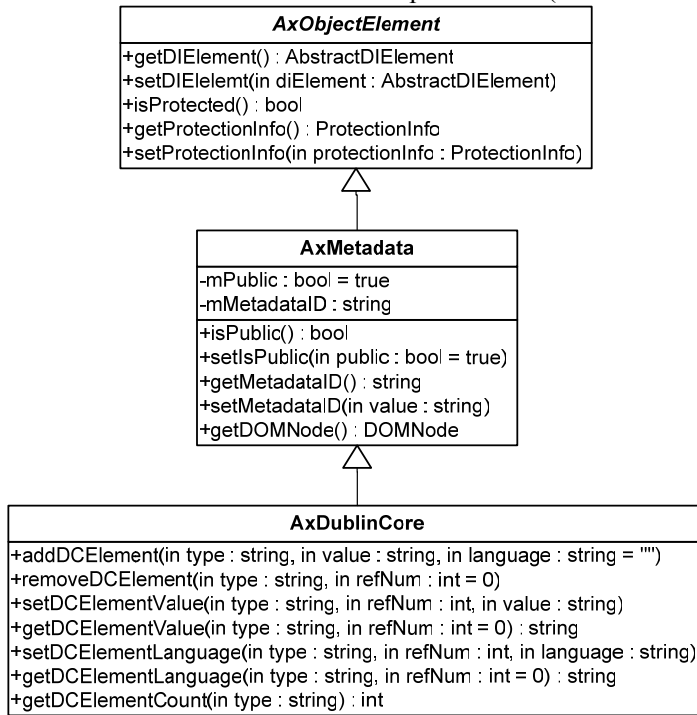
Object History Management

+getHistoryOfVersion(in version:int) : DOMNode
gets the history of a version as a DOM Node

8.2.2.3 AxDublinCore

Class AxDublinCore allows to manage a Dublin Core descriptor:

DE3.1.2A – Framework and Tools Specification (General and Model)



Example of use:

```

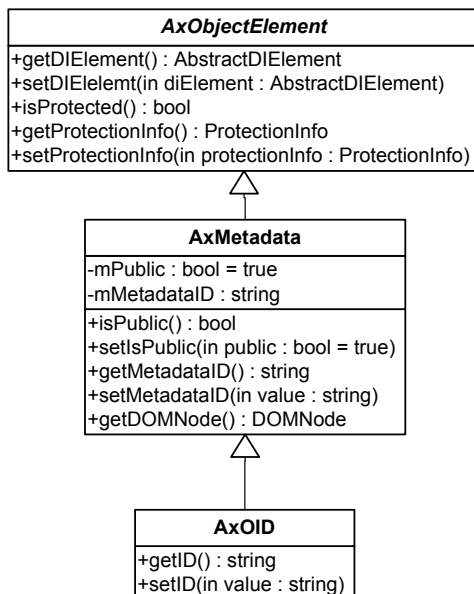
AxDublinCore aDC;

if (aDC.getDCElementCount ("creator")==0)
    aDC.addDCElement ("creator", "Mozart");
else
    aDC.setDCElement ("creator", 0, "Mozart");

string creator=aDC.getDCElementValue ("creator");
  
```

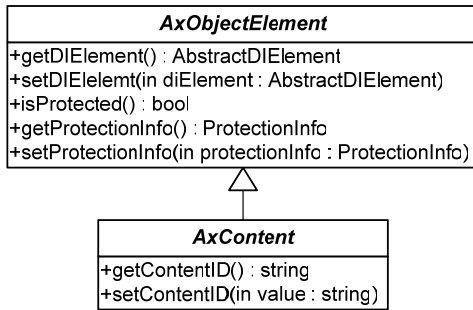
8.2.2.4 AxOID

Class AxOID embeds the AXOID identifier.

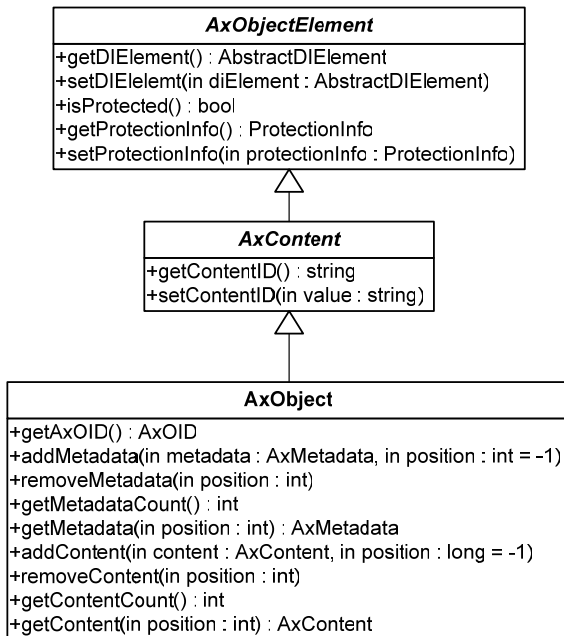


8.2.2.5 AxContent

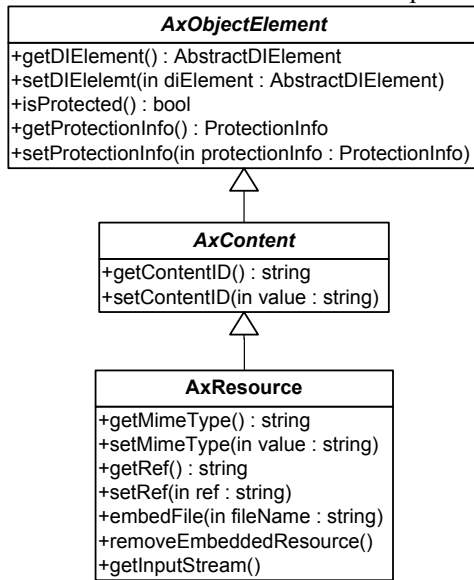
Abstract class *AxContent* represents content to be stored in AXMEDIS objects, it is specialized in *AxObject* or *AxResource*.

**8.2.2.6 AxObject**

Class *AxObject* represents an AXMEDIS Object, it can contain any number of metadata and any number of content and it is identified by an *AxOID*.

**8.2.2.7 AxResource**

Class *AxResource* represents any digital resource identified with a mime type, it can be an image, a document, an audio etc.



8.3 XML Loader and Saver (DSI)

Module Profile		
XML Loader and Saver		
Executable or Library(Support)	Support Library	
Single Thread or Multithread	Single Thread	
Language of Development	C++	
Responsible Name	Davide Rogai, Andrea Vallotti	
Responsible Partner	DSI	
Status (proposed/approved)	proposed	
Platforms supported	All...	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces-C++	v. 2.6.0	Apache Licence 2.0

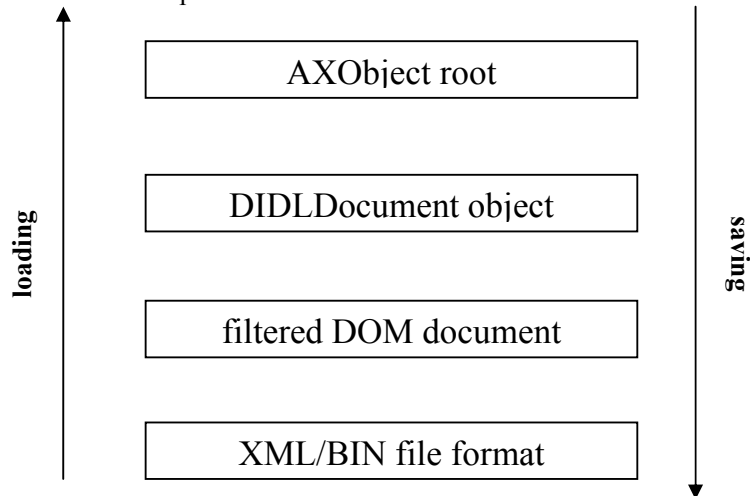
DE3.1.2A – Framework and Tools Specification (General and Model)

XML Loader is the tool by which a client (Player, Authoring tool, Formatting/Compositional process engine, etc...) can obtain a software representation of an AXMEDIS object from an XML (text or binary) file which represents such object.

The XML loader is structured as the concatenation of different transformations:

1. from XML file to filtered DOM
2. from filtered DOM to MPEG21 DIDL memory object tree
3. from MPEG21 DIDL to AXMEDIS object tree

It is natural to think the XML Saver performs the reversed transformations.



Loading steps

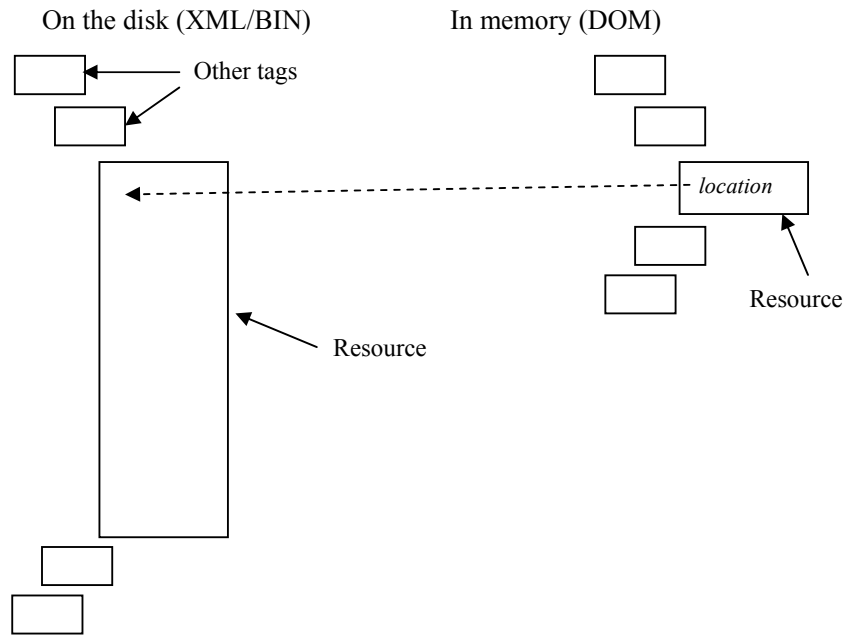
The first step is processing the XML or BIN format of an AXMEDIS object. The resources of the object could be too big to allocated as a whole in memory; filtering is the only possibility in order to get the benefits of a DOM in memory of the AXMEDIS XML format, without the occupation of (potentially) GBytes of data (e.g. a MPEG2 video resource of a movie). The DOM implementation expose a node filtering feature that can reject a node after the DOM builder has loaded it; this is unacceptable for the loader purpose, because the resource could be only removed after it has occupied the whole memory and the loader has caused a fatal error on the system.

The solution is to intercept the “resource” tag occurrence before it is loaded in memory; something that the SAX interface can manage.

It is important to understand that a reference must not be loaded as a whole, but it must be inserted in the DOM structure as an equivalent entity.

In the DOM the resource includes a way to retrieve the content stream still located on the file system.

DE3.1.2A – Framework and Tools Specification (General and Model)



The new resource element must redirect access to the effective content location (stream to the file/storage position) during access functions (play, adapting, copying...).

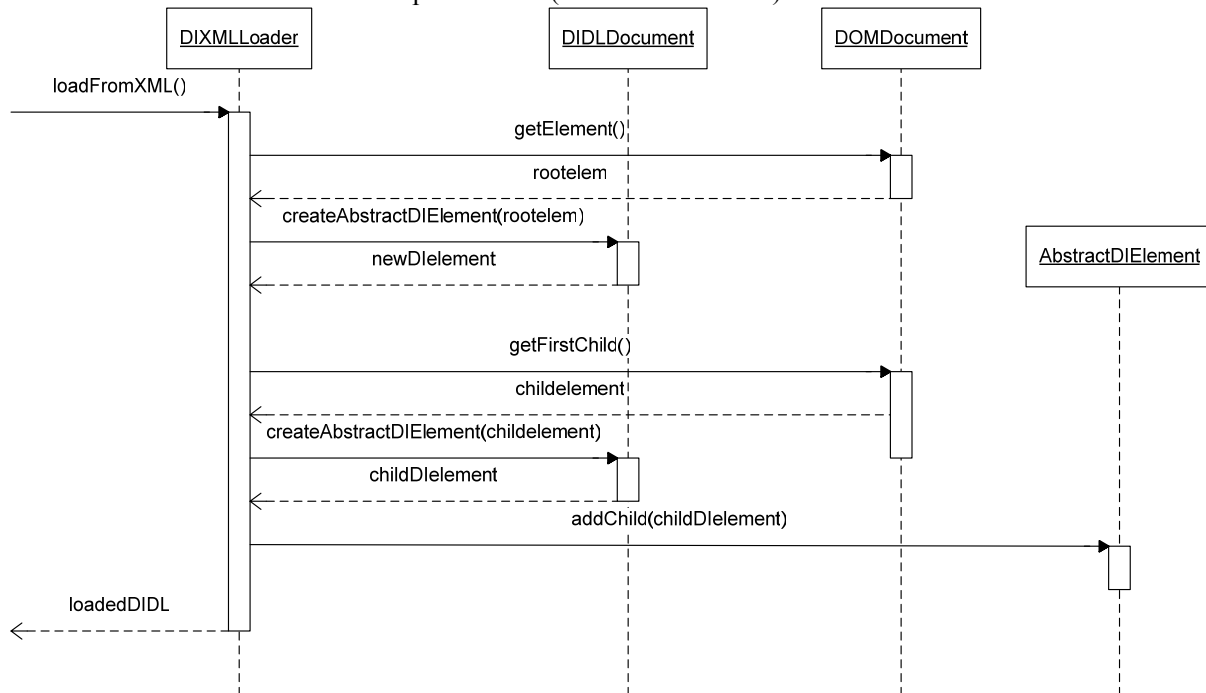
The IPMP Content of an IPMPItem (it is used to protect content inside AXMEDIS objects) is treated in the same way; it could be a big amount of data and it will be not loaded in the DOM. The part in the DOM that refer to a stream on the file will be integrated if requested by the user actions.

The hierarchy of DIDLElement objects is built on the basis of the DOM created from the XML file.

The class responsible for loading an object from XML file is XMLDILoader. The class provide only one method that is *loadFromXML(string path)* that returns a DIDLDocument.

In the following a sequence diagram of the loading interactions among different parts of the action is depicted. The loading is invoked on the DIXMLLoader passing to it the path of the XML file that has to be loaded. Then a DOMDocument is created processing the file and browsed by the loader: inside the loader class there is the logic to fetch elements in the DOMDocument and to append in the proper position to the DIDLDocument hierarchy of AbstractDIElement objects. The DIDLDocument act also as the factory of AbstractDIElement objects; it is able to process a DOMELEMENT node of the DOMDocument in order to create a new AbstractDIElement. The loader uses this features and it is responsible to fill pointers to related elements in the hierarchy.

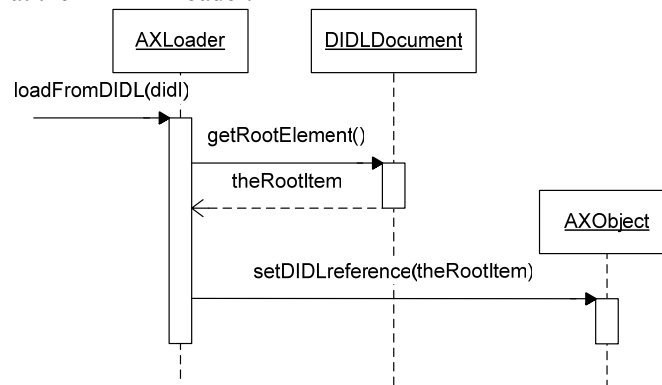
DE3.1.2A – Framework and Tools Specification (General and Model)



The second phase of loading AXMEDIS objects concerns the construction of AXMEDIS hierarchy based on the MPEG21 DIDL one.

Another class plays the role of object loader (entity by entity), but is related to the MPEG21 document root (a DIDLDocument object).

Since the information complexity is already modelled by MPEG21 data model, the AXLoader behaviour appears much simpler than the DIXMLLoader.



Saving steps

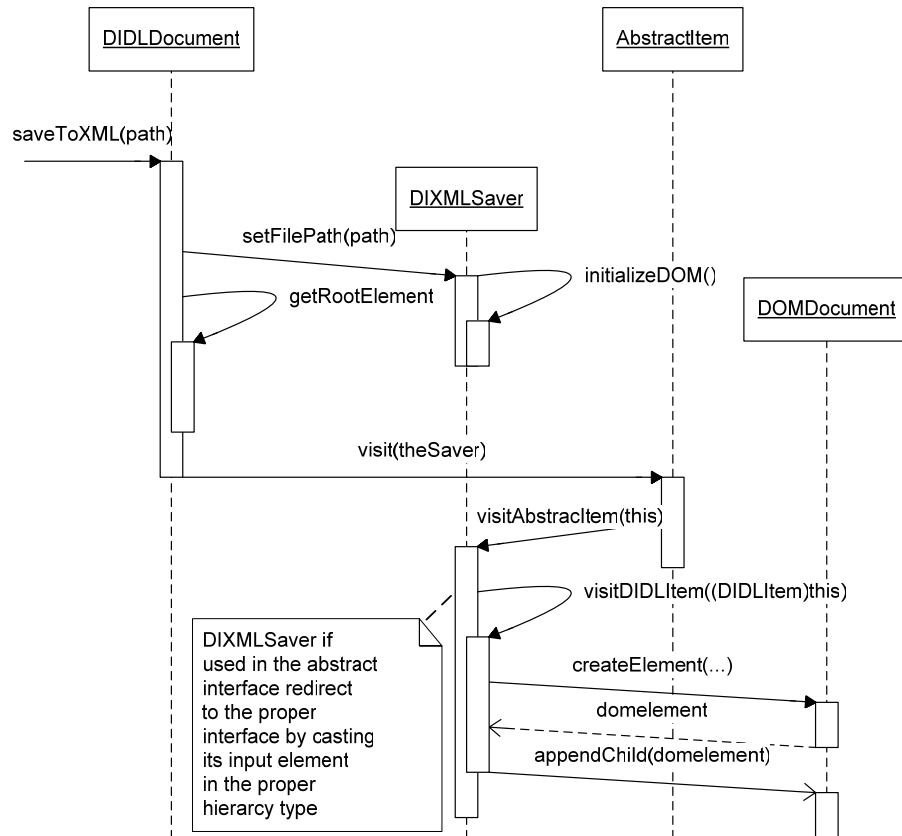
Since the AXMEDIS hierarchy, which originates in a AXObject reference, is strictly dependent from the AbstractDIElement object referred from its children. This two hierarchies of object are forced to be consistent with each other, so the save process at AXMEDIS level consists in save the DIDLDocument connected to the AXMEDIS Object. The real information regarding the AXMEDIS Object are stored at MPEG21 DIDL level and only rendered at AXMEDIS level by group classes.

The DIDLDocument save process is based on the visitor pattern: in fact the DIXMLSaver realizes the AbstractDIElementVisitor, because it has to navigate over all the elements referenced in the hierarchy.

The DIXMLSaver has to implements all the visitor interfaces since it is responsible to visit any element of the hybrid (DIDL/IPMP) hierarchy. The DIXMLSaver uses a DOMDocument to write data gathered during the visitation of the AbstractDIElement objects.

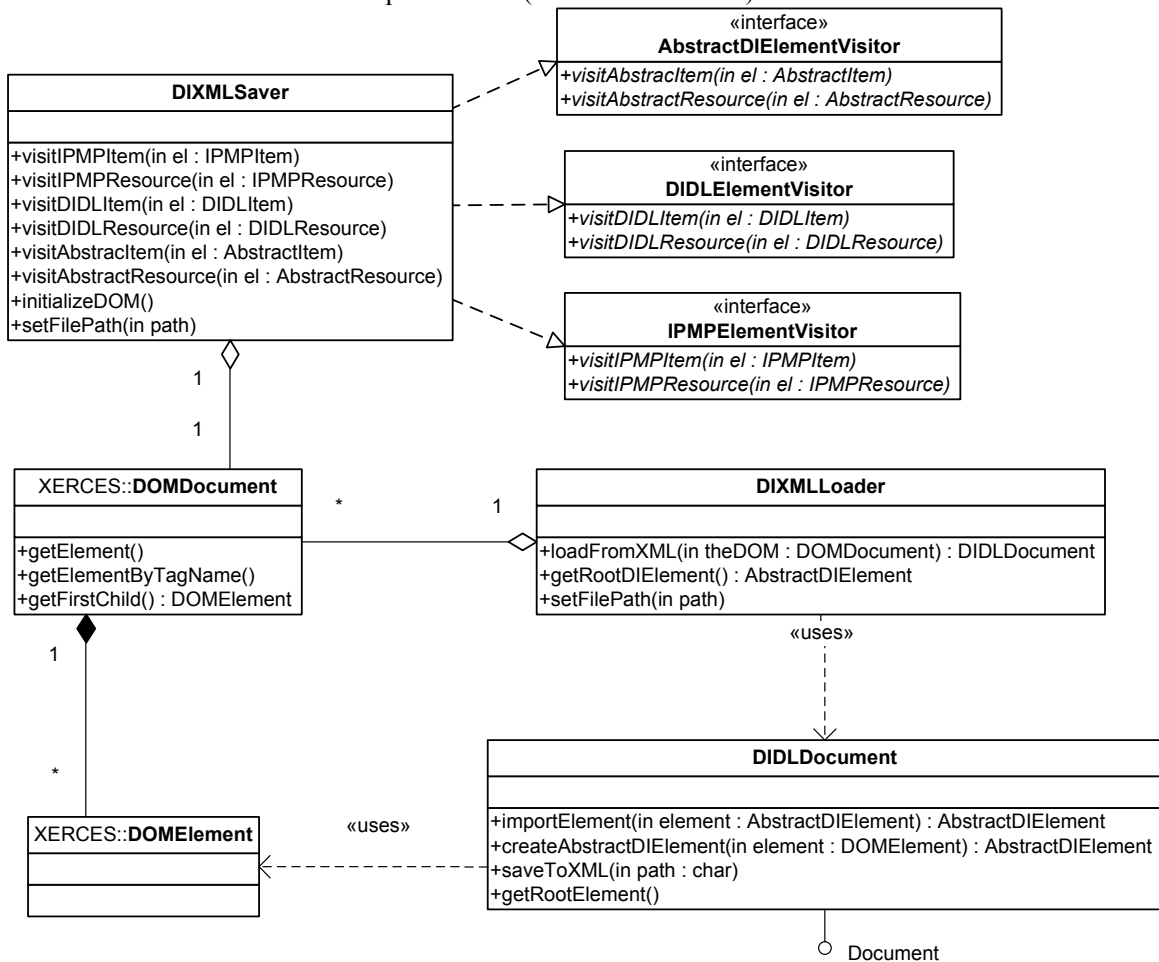
DE3.1.2A – Framework and Tools Specification (General and Model)

In the following a sequence diagram highlights delegation between objects and visitor interfaces.

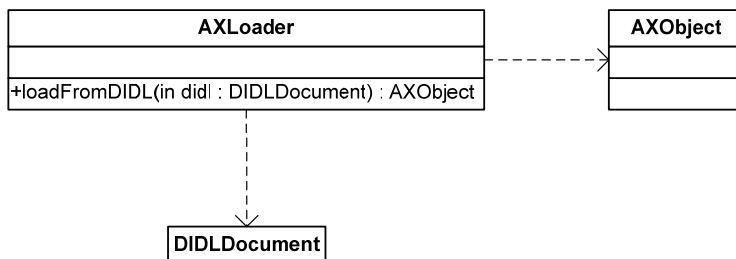


It follows the static diagram that shows relationships among classes at Digital Item level. The Saver is a visitor for all the hierarchies provided by the MPEG21 Data model. The Loader uses the DIDLDocument as a factory to creates AbstracDIElement objects. Both classes are related to a DOM in order to perform read/write operations on an XML file.

DE3.1.2A – Framework and Tools Specification (General and Model)



Also in the static diagram the complexity is reduced for the AXLoader class, as depicted below.



8.3.1.1 Xerces-C++

Xerces-C++ is a validating XML parser written in C++. Xerces-C++ makes it easy to give your application the ability to read and write XML data. A shared library is provided for parsing, generating, manipulating, and validating XML documents, including reference solver/integrator for e.g. combining multiple files.

Source code, samples and API documentation are provided with the parser. Xerces is available for several platforms among them Windows OS, and Red Hat Linux.

Xerces implements two different APIs for XML parsing: SAX and DOM. Both of them define the logical structure of documents and the way a document is accessed and manipulated. SAX was first adopted by Java programmers while DOM is a W3C specification.

License

Xerces can be used with no charge, and in a royalty-free way. No obligation to redistribute the source code. The only obligation is to redistribute a copy of the license notice.

Installation

The compiled libraries `-.lib .dll-` and the source files `-.c .hpp-` are available for download from <http://xml.apache.org/xerces-c/download.cgi>. No installation is needed though they insist on verifying the integrity of the downloaded files by using digital signatures.

Example of use in C++: DOM example

```
#include <xercesc/dom/DOM.hpp>
```

The header file `<dom/DOM.hpp>` includes all the individual headers for the DOM API classes.

The DOM class names are prefixed with "DOM" (if not already), e.g. "DOMNode". The intent is to prevent conflicts between DOM class names and other names that may already be in use by an application or other libraries that a DOM based application must link with.

```
DOMDocument* myDocument;
```

```
DOMNode* aNode;
```

```
DOMText* someText;
```

Applications would use normal C++ pointers to directly access the implementation objects for Nodes in C++ DOM.

```
DOMNode* aNode;
```

```
DOMNode* docRootNode;
```

```
aNode = someDocument->createElement(anElementName);
```

```
docRootNode = someDocument->getDocumentElement();
```

```
docRootNode->appendChild(aNode);
```

8.4 AXMEDIS Object Preprocessor and Postprocessor (EPFL)

Module Profile	
AXMEDIS Object Preprocessor and Postprocessor	
Executable or Library(Support)	
Single Thread or Multithread	
Language of Development	
Responsible Name	
Responsible Partner	
Status (proposed/approved)	
Platforms supported	

DE3.1.2A – Framework and Tools Specification (General and Model)

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

Tools for transforming XML into binary and viceversa, for transforming external references in internal and viceversa, etc.

8.4.1 XML to/from BIN (EPFL)

Using user-defined tags in text format make XML files oversized; this is known as XML verbosity. It is a characteristic of mark-up languages like HTML or text languages in general. There is redundant information in XML files like blank spaces or attribute names. The overhead of textual XML can be overcome by compressing and converting the text file into a binary file. Storage of big files is not a concern nowadays because high capacity hard disks are not expensive. Nevertheless, the reduction of the file size is especially important when the file is transmitted through channels where the bandwidth is a constraint e.g. wireless or Internet. The smaller is the file the less bandwidth is consumed.

8.4.1.1 BIM

The use of BiM is proposed. Other compressors like Xmill, XMLZip, or WBXML achieve the same compression ratios than BiM does but neither of them fulfils all the BiM requirements: flexible updating, random access, high compression, light decoding, and parsing at binary level.

Module Profile		
XML to/from BIN		
Executable or Library(Support)	Support Library	
Single Thread or Multithread	Single Thread	
Language of Development	C/C++	
Responsible Name		
Responsible Partner		
Status (proposed/approved)	Proposed	
Platforms supported	Windows / Linux	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)

DE3.1.2A – Framework and Tools Specification (General and Model)

File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
MPEG-7 Experimentation Model may be used	XM (version 5.5 ?)	Free, but patent pool for BiM still to be investigated

Overall Module Description

BiM (Binary Format for MPEG-7) is the MPEG-7 XML compression standard. BiM encoder and decoder can be used with any XML language and they have been already tested on XHTML and SVG. BiM is a schema-based compression technique. It exploits the knowledge of the XML schema to compress the file. In BiM, the XML schema has to be known by the encoder and the decoder. The elements inside the XML file are compressed with other encoders based on redundancy elimination techniques. The standard specifies a way to plug-in other dedicated encoders that can be better to compress certain elements of the XML file. BiM does a partial validation of the file while encoding so that the decoder has to do less processing to validate the file. Type information is associated to the XML components during the encoding process to achieve a bigger compression. At the decoder side, XML values are received in type format so that they are quicker to process than text XML values. As XML, BiM can be robust against versioning and evolution of the XML language. An old decoder might be able to partially decode a new version of binary XML file. Yet if no compatibility is needed, higher compression ratios can be achieved. BiM is able to decompose a big XML file in smaller parts and then encode each part independently. The BiM decoder is able to extract information from the binary XML parts without having them all available. This feature is useful when updating only the part that has changed instead of reloading the entire file. It is also a big advantage over only-redundancy encoders because in streaming applications it allows the decoder to start decoding before it has received the complete file. It is possible to navigate through the XML tree architecture in binary format without the necessity of decoding the whole XML binary file. By navigating through the binary format of the file, the encoder does not have to decode the entire file but only the sub-tree or element in which the user is interested. This turns into a save of memory, of processing, and time because not all the file has to be decoded to access a part of it.

8.4.1.2 Xmill

XMill is a special-purpose compressor for XML data that typically achieves twice the compression rate over existing compressors, such as gzip.

XMILL is an open source program delivered under a BSD License, it is written in C++, and it is available for all 32-bit MS Windows (95/98/NT/2000/XP), and all POSIX (Linux/BSD/UNIX-like OSes)

Similar to gzip, XMill is a command-line tool that works on a file-by-file basis. A given file with extension '.xml' is compressed into a file with extension '.xmi'. Any other file without extension '.xml' is compressed into a file by appending extension '.xm'. Reversely, the original file is obtained by replacing extension '.xmi' with extension '.xml' or by removing extension '.xm'. Alternatively, the user write the output to the standard output and optionally read from the standard input.

Underlying Technique

XMill is based on a grouping strategy that groups and compresses text items together based on their semantics. For example, a sequence of <Person> elements (with <Name>, <Age>, <Shoesize>, ... elements) in an XML document could be rearranged by grouping all names, all ages, and all shoe sizes together. This will typically lead to higher compression ratio, since each group will contain text items with high similarities.

The default grouping strategy is by considering the parent label of the text item. Even though this works well, there are cases when a label has different meanings in different parts of the document (e.g. <Title> in <Person> has a different meaning from the <Title> in <Book>) or when different labels have the same meaning (e.g. a <ChildName> in <Person> contains person name like <Name>).

Therefore, XMill provides a powerful regular path expression language for grouping text items with respect to their meaning. Each text item is reachable over a 'path' of labels from the root of the XML document. For example, the social security of an employee of a company might be stored in the 'ssno' attribute of element '<Employee>' in element '<Company>' in the root element '<Root>'. Hence, the path to the text item is /Root/Company/Employee/@ssno. After the grouping in containers, conventional compressors, such as gzip are applied to the containers and will exploit those similarities. Since the number and size of containers grows with the size of the XML file, a memory window mechanism is implemented: After the overall size of the containers reaches a certain user-specified memory window, the containers are compressed and stored in the output file. The compressed content of the memory window is called a 'run'. After the 'run' is stored in the output file, the containers are filled with data again.

In addition to path expression, the user can also specify how to "pre-compress" the specific text item. For example, the user might want to replace the 'Age' string by its binary integer representation. Or more complex, an IP number might be replaced by four bytes.

XMill allows the user to specify additional "user compressors" to pre-compress the text items before it is stored in the containers. Note that the gzip compression is still applied to the containers afterwards.

XMill provides an interface for writing own user compressors in C++. This is particularly useful for domain-specific data, such as DNA sequences, 3D coordinates, etc.

8.4.1.3 BinXML

BinXML is the Binary XML transport and processing format adopted by MPEG-7, TV-Anytime and ARIB consortia. BinXML provides a high XML data compression rate, which helps saving memory, storage space, or bandwidth in an XML architecture. BinXML provides high processing speed of the encoded XML, with very low hardware requirements in memory and CPU power, contrarily to textual XML data.

BinXML is an open framework which accepts authentication, signature or error correctness tools (third party softwares). These tools can be currently applied, to whole BinXML documents after the encoding, and in the future, to XML fragments when the streaming functionality will be released.

BinXML Software Development Kit is currently available for Microsoft Windows and Linux based systems. The BinXML starter kit tryout is fully functional production software. It is restricted by a **30-days** key and contains the minimal set of tools to allow encoding and decoding XML files.

BimXML is a proprietary implementation of MPEG-7 BiM (supposed to become soon MPEG-B) and so it implements all the BiM capabilities: pre-parsed format, typed format, schema-oriented, etc.

Using BinXML to encode an XML file

The following C++ example will encode “input_file.xml” to “output_file.out” if we would execute it. The file “license.dat” allows using the BinXML library for 30 days. The file “ExManagerConfig.xml” contains the encoder configuration. Since BinXML is schema-oriented it needs to know the schema we are using. The schema information is passed to the library through the “ExManagerConfig.xml” file. The SDK provides a utility program that changes the format of our XML schema to a format that the library understands. Once the schema has been converted, by using the utility program, it is referenced in the “ExManagerConfig.xml” configuration file.

```
char* szSource = "input_file.xml";
char* szDestination = "output_file.out";

int iRet = 0;
ExampleErrorHandler* pErrorHandler = new ExampleErrorHandler();
IEwManager* pManager;

EwManagerFactory::registerLicenseFile("license.dat");

// Create and setup the manager
pManager = EwManagerFactory::createManager("EwManagerConfig.xml",
pErrorHandler);

EwFileAUHandler* pAUHandler = new EwFileAUHandler(szDestination);
pManager->setAUHandler(pAUHandler);

xercesc::ContentHandler* ch = pManager->getContentHandler();
EwXMLInput* pInput = new EwXMLInput(pErrorHandler);
pInput->setContentHandler(ch);

// encode the file !
pInput->parse(szSource);

// Cleanup
delete pAUHandler;
delete pInput;
EwManagerFactory::destroyManager(pManager);
```

8.4.2 References In/Out Resolver/Integrator (EPFL)

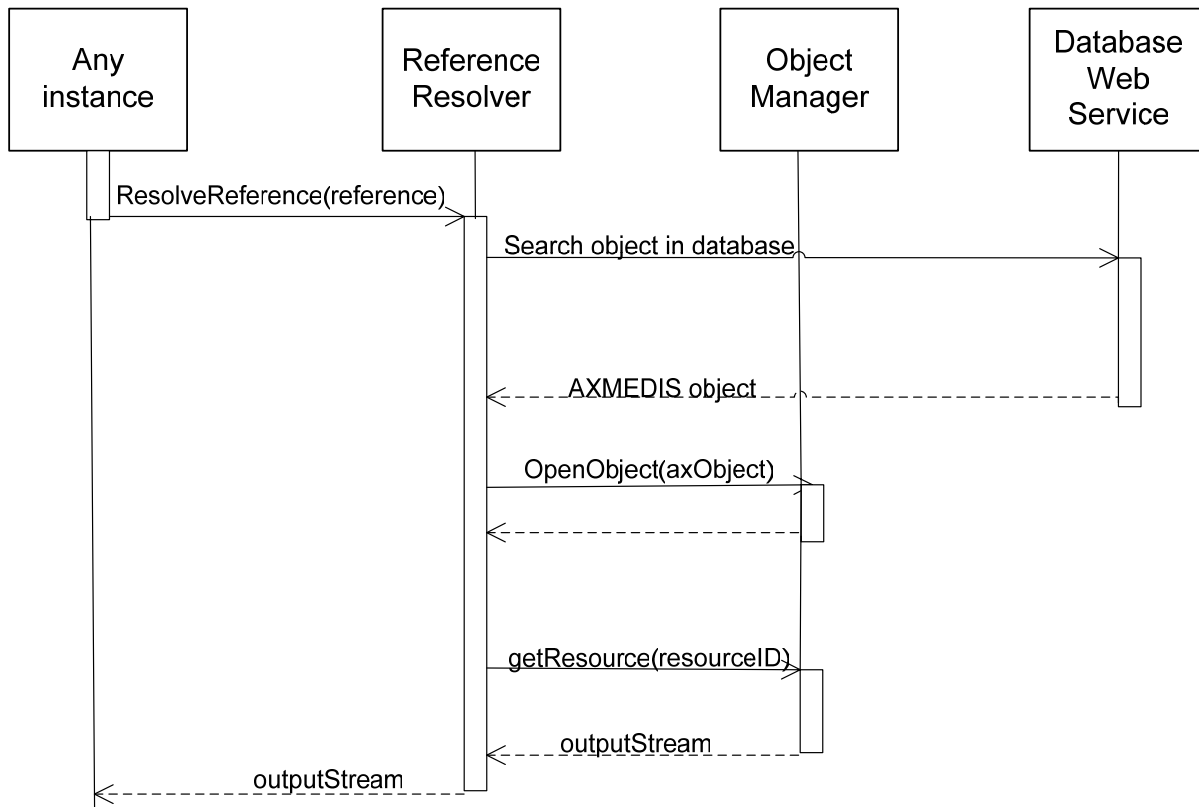
Module Profile		
References in/OUT Resolver/Integrator		
Executable or Library(Support)		
Single Thread or Multithread		
Language of Development	C++	
Responsible Name		
Responsible Partner		
Status (proposed/approved)	proposed	
Platforms supported	Windows, Linux	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a

DE3.1.2A – Framework and Tools Specification (General and Model)

		section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces-C++		Freedom to use and to distribute without paying royalties nor distributing source code

In this flow diagram an instance, a player for example, asks to resolve a reference. We assume that the object referenced is a resource contained inside an AXMEDIS object located in the AXMEDIS database. The caller instance uses the function `ResolveReference(reference)` on the `ReferenceResolver`. The `ReferenceResolver` uses the Database Web Service to find and download the AXMEDIS object inside of which there is the target resource. Once the AXMEDIS object is downloaded, the `ReferenceResolver` uses the `ObjectManager` to extract the resource from the AXMEDIS Object. Finally, the resource is returned to the calling instance by means of a pointer or reference to an `outputStream` interface.

If the referenced object is not in the local file system, the Reference Resolver uses the Axmedis Database Web Service to localize and download the object.



DE3.1.2A – Framework and Tools Specification (General and Model)

The following flow diagram shows how a reference integration could take place. In this case we assume that some instance wants to insert in the AX Object2 a resource that is located inside of the AX Object1. The ReferenceIntegrator locates first the AX Object1 by using the Database Web Service. Then it extracts the target resource from the AX Object1 by using the Object Manager1. Finally it uses the Object Manager2 to add the resource into the AX Object2.

