Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE3.1.2.3.6
Specification of AXMEDIS Content Processing Area – update of DE3.1.2.2.6
Updated version at M48

Version: 1.8
Date: 12.11.2008
Responsible: Ivan Bruno (DSI) (verified and approved by coordinator)

Project Number: IST-2-511299 Project Title: AXMEDIS Deliverable Type: report Visible to User Groups: yes Visible to Affiliated: yes Visible to the Public: yes
Deliverable Number: DE3.1.2.3.6 Contractual Date of Delivery: M48 Actual Date of Delivery: 12/11/2008 Title of Deliverable: Specification of AXMEDIS Content Processing Area Work-Package contributing to the Deliverable: WP3.1 Task contributing to the Deliverable: WP3, WP2 Nature of the Deliverable: report Author(s): DSI

Abstract: this part includes the specification of components, formats, databases and protocol related to the AXMEDIS Framework area AXMEDIS Content Processing Area, the formatting Engine as module and component for AXCP and the specification of the Javascript classes for the AXMEDIS Script languge.
Keyword List: Content production, Javascript, Adaptation tools, formatting, composition, transcoding

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**

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-fulfillment of any of his obligations in respect of this License.

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

The full AXMEDIS specification document has been decomposed in the following parts:

DE number	Deliverable title	responsible
DE3.1.2.3.1	Specification of General Aspects of AXMEDIS framework AXMEDIS-DE3-1-2-3-1-Spec-of-AX-Gen-Asp-of-AXMEDIS-framework	DSI
DE3.1.2.3.2	Specification of AXMEDIS Command Manager AXMEDIS- DE3-1-2-3-2-Spec-of-AX-Cmd-Man	DSI
DE3.1.2.3.3	Specification of AXMEDIS Object Manager and Protection Processor AXMEDIS-DE3-1-2-3-3-Spec-of-AXOM-and-ProtProc	DSI
DE3.1.2.3.4	Specification of AXMEDIS Editors and Viewers AXMEDIS-DE3-1-2-3-4-Spec-of-AX-Editors-and-Viewers	DSI
DE3.1.2.3.5	Specification of External AXMEDIS Editors/Viewers and Players AXMEDIS-DE3-1-2-3-5-Spec-of-External-Editors-Viewers-Players	DSI
DE3.1.2.3.6	Specification of AXMEDIS Content Processing AXMEDIS-DE3-1-2-3-6-Spec-of-AX-Content-Processing	DSI
DE3.1.2.3.7	Specification of AXMEDIS External Processing Algorithms AXMEDIS-DE3-1-2-3-7-Spec-of-AX-External-Processing-Algorithms	FHGIGD
DE3.1.2.3.8	Specification of AXMEDIS CMS Crawling Capabilities AXMEDIS-DE3-1-2-3-8-Spec-of-AX-CMS-Crawling-Capab	DSI
DE3.1.2.3.9	Specification of AXMEDIS database and query support AXMEDIS-DE3-1-2-3-9-Spec-of-AX-database-and-query-support	EXITECH
DE3.1.2.3.10	Specification of AXMEDIS P2P tools, AXEPTool and AXMEDIS AXMEDIS-DE3-1-2-3-10-Spec-of-AXEPTool-and-AXMEDIA-tools	DSI
DE3.1.2.3.11	Specification of AXMEDIS Programme and Publication tools AXMEDIS-DE3-1-2-3-11-Spec-of-AX-Progr-and-Pub-tool	UNIVLEEDS
DE3.1.2.3.12	Specification of AXMEDIS Workflow Tools AXMEDIS-DE3-1-2-3-12-Spec-of-AX-Workflow-Tools	UR
DE3.1.2.3.13	Specification of AXMEDIS Certifier and Supervisor and networks of AXCS AXMEDIS-DE3-1-2-3-13-Spec-of-AXCS-and-networks	DSI
DE3.1.2.3.14	Specification of AXMEDIS Protection Support AXMEDIS-DE3-1-2-3-14-Spec-of-AX-Protection-Support	UPC
DE3.1.2.3.15	Specification of AXMEDIS accounting and reporting AXMEDIS-DE3-1-2-3-15-Spec-of-AX-Accounting-and-Reporting	EXITECH

1.1 This document concerns

This document is focused on producing an update of the specification to be adopted as a baseline in the specification tasks of the different work packages that will be developed in XXXX months of the project.
The part C of the specification deals with problems related to the Content Production and more in general with content processing, therefore the specification of the Content Processing and tools are reported.
Specification Part C is structured in 3 main sections dealing with different aspects of content production:

- **AXMEDIS Content Processing Area** (under responsibility of DSI). This section describes the goals, the activity and tools of AXMEDIS Content Processing Area. This area copes with the problem of automatic content production, adaptation and protection of AXMEDIS object and their publication on a P2P environment (AXEPTool). The proposed solution is based on rules that include a procedural description using the Javascript language (script) and a Javascript engine for the their execution. The JavaScript engine is derived from SpiderMonkey by Mozilla. To cope with the amount of needed resources (computational, time, etc…) during the content processing activity, a distributed environment has been defined and based on GRID computing. The AXMEDIS Content Processing Area specification is then further illustrated by means of UML diagrams, scenarios, tables, GUI definitions and design, definition of a Grid communication protocol and the XML schema for rules.
- **AXMEDIS Content Format Engine** (under responsibility of DSI). This section describes the goals, of the AXMEDIS Content Format Engine. This area copes with the formatting functionalites that are used by the AXCP GRID and by format tools (*Template Editor and Selector*, *Style Editor and Selector*, *Style Optimizer*) integrated within the AXMEDIS SMIL Editor. The goal of the Content Format Engine is integrating digital resources, contained within an AXMEDIS Object, in a multimedia presentation (based on the SMIL language) suitable for the final user.
- **AXMEDIS Data Types and functions for JavaScript Engine**. In this section how to use C++ classes and functions in the Spidermonkey Javascript engine are described. This mechanism has been used to wrap the set of classes (methods and attributes) and functions to be used with the Javascript code The current set of classes and functions are also reported.

1.2 List of Modules or Executable Tools Specified in this document

Module/tool Name	Module/Tool Description and purpose, state also in which other AXMEDIS area is used	Standards exploited if any
AXCP Rule Model	It models the AXCP rule and provides loader/saver functionality to load and save rules	AXMEDIS common and XERCES v2-6 libraries
AXCP Rule Editor	The application is an executable GUI for creating, editing, saving, testing and activating AXCP rules Other AXMEDIS Modules include ▪ AXCP Rule Model ▪ AXCP Rule Executor for testing rules ▪ AXMEDIS Configuration Manager for application ▪ AXMEDIS Plugin Manager for Rule Editor Workflow plugin and AXCP Plugins ▪ AXMEDIS Content Processing Plugins ▪ AXMEDIS Selection Editor	It exploits the AXCP Rule Model AXMEDIS Framework, AXCP Rule Executor and XERCES v2-6 libraries
AXCP Rule Engine	It consists in: ▪ AXCP Rule Scheduler: the application in charge to manage AXCP rules and schedule their execution in the GRID environment ▪ AXCP Rule Executor: it is an instance of the Rule executor in the GRID environment and executes AXCP rules ▪ AXMEDIS Configuration Manager for application ▪ AXMEDIS Plugin Manager for Rule Scheduler Workflow and AXCP Plugins ▪ AXMEDIS Content Processing Plugins	It exploits the AXCP Rule Model AXMEDIS Framework, AXCP Rule Executor and XERCES v2-6 libraries

AXMEDIS Content Format Engine	It provides content formatting functionalities that are exploited by the AXCP Rule Engine and by other authoring tools of the AXMEDIS framework. It allows both automatic formatting and interactive access to base components used by the module.	AXMEDIS common and XERCES v2-7 libraries
AXMEDIS Data Types and functions for JavaScript Engine	It consists in several modules. Each module provides a JS class for the JS Engine based on Spidermonkey derived by classes of the AXMEDIS Framework.	Libraries/standards used in the development of AXMEDIS Framework

1.3 List of Formats Specified in this document

Format Name	Format Description and purpose, state also in which other modules is used	Standards exploited if any
AXCP Rule	It is an xml description (xsd schema) of rule. It is used in: ▪ AXCP Rule editor ▪ AXCP Rule Scheduler ▪ AXCP Rule Executor ▪ AXPnP Editor	XERCES v2-6 and the AXMEDIS common libraries
AXCP Executor Profile	It is an xml description (xsd schema) of executor profile, in terms of capabilities and available resources It is used in: ▪ AXCP Rule Scheduler ▪ AXCP Rule Executor	XERCES v2-6 and the AXMEDIS common libraries
Resource descriptor	It is an xml description (xsd schema) of digital resources. It is used in: ▪ Content Formatter Engine	XERCES v2-7 and the AXMEDIS common libraries
Template descriptor	It is an xml description (xsd schema) of format templates. It is used in: ▪ Content Formatter Engine	XERCES v2-7 and the AXMEDIS common libraries
Style descriptor	It is an xml description (xsd schema) of format style-sheets. It is used in: ▪ Content Formatter Engine	XERCES v2-7 and the AXMEDIS common libraries
User Profile descriptor	It is an xml description (xsd schema) of user profile. It is used in: AXCP Engine	MPEG 21 (Partially)
Device Profile descriptor	It is an xml description (xsd schema) of device profile. It is used in: AXCP Engine	MPEG 21 (Partially)
Network Profile descriptor	It is an xml description (xsd schema) of user profile. It is used in: AXCP Engine	MPEG 21 (Partially)

1.4 List of Databases Specified in this document

Database Name	database Description and purpose, state also in which other AXMEDIS area is using	Standards exploited if any
AXMEDIS Database	Accessed using the AXOM loader and saver tools	

1.5 List of Protocols Specified in this document

Protocol Name	protocol Description and purpose, state also in which other modules is used	Who is the master and who is the slave	Standards exploited if any
AXCPGridRuleEngine	It is a ASCII based set of commands used as communication support in the GRID environment. It is used in: ▪ AXCP Rule Scheduler ▪ AXCP Rule Executor	AXCP Rule Scheduler is the master and AXCP Rule Executors are slaves	

2 General Use Cases and scenarios

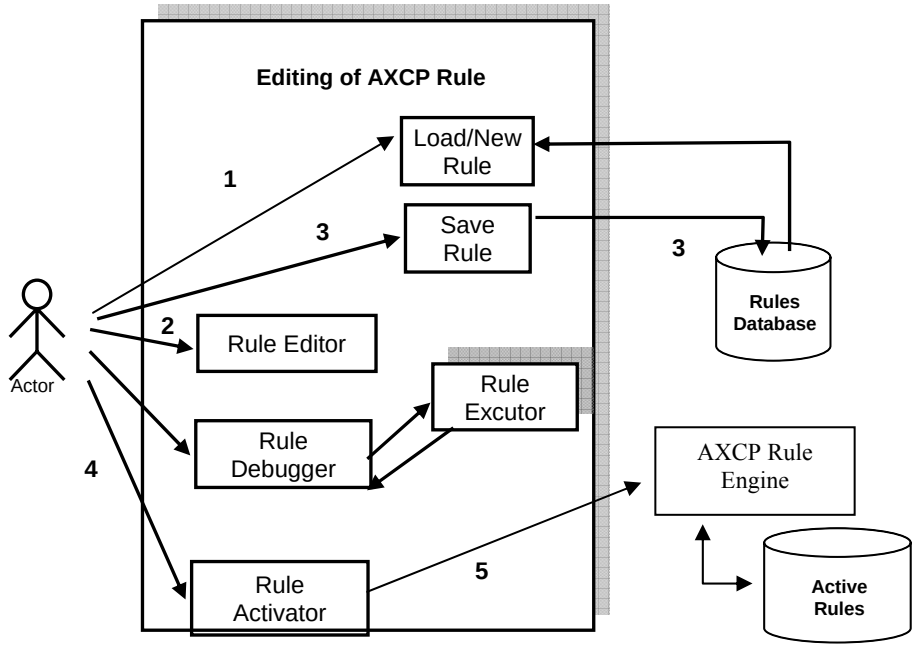
The use case for the AXMEDIS Content Processing Area includes:

- Use case 2.1: Editing and Activation of an AXCP rule
- Use case 2.2: Execution of a rule in the **AXCP Rule Engine**

]

2.1 Use Case: Editing and Activation of an AXCP rule

The use case shows how AXCP rules are produced.

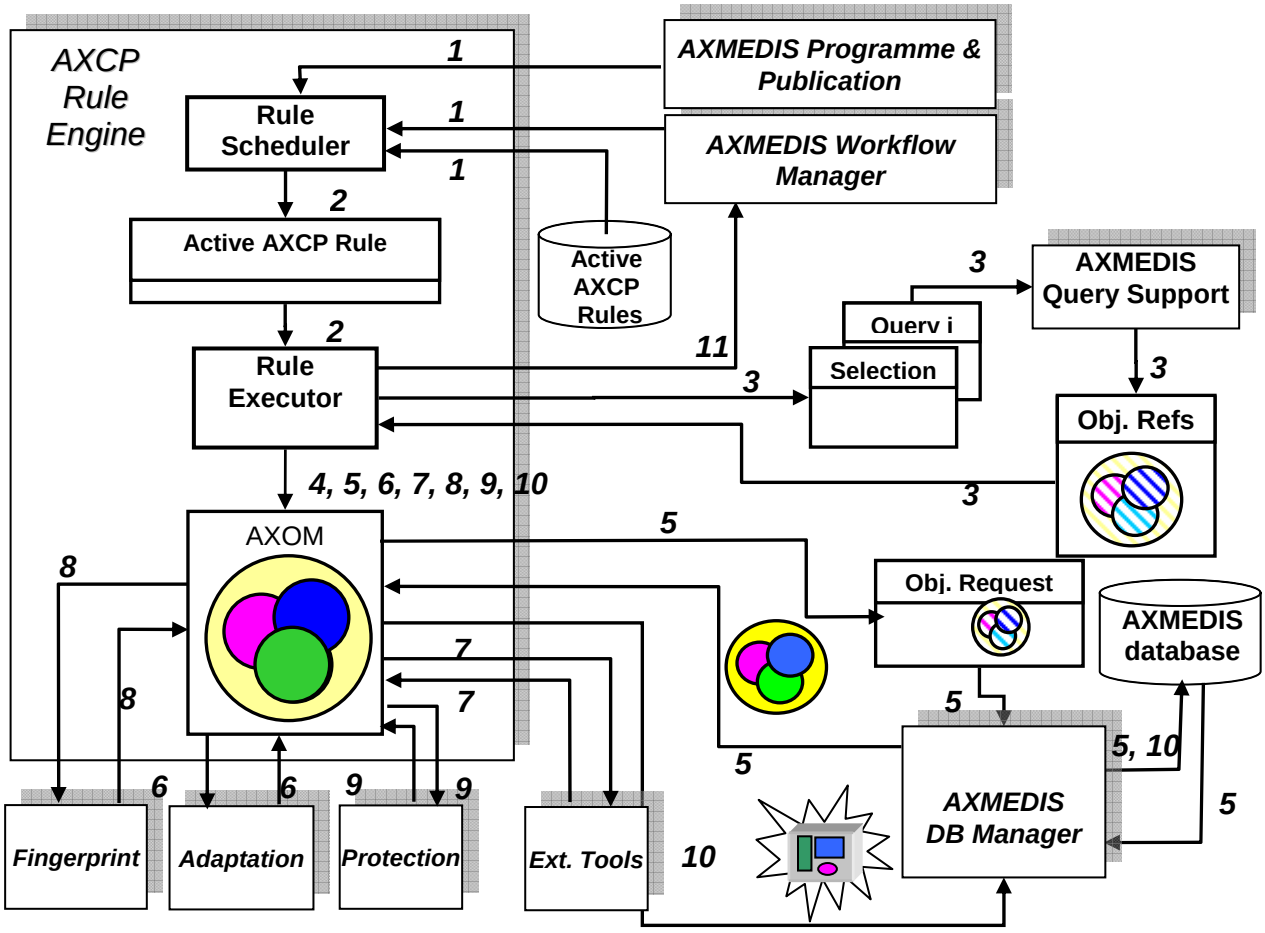


Scenario description for the activation and modification of an existing rule:

1. The actor loads an existing AXCP rule from the rules database or starts with a new rule
2. The actor edits the rule by the AXCP Rule Editor
3. The actor saves the new rule into the rules database
4. The actor activate the rule
5. The rule is sent to rules repository of the AXCP Rule Engine).

2.2 Use Case: Execution of a rule in the AXCP Rule Engine

The use case shows the execution of a rule in the **AXCP Rule Engine**.



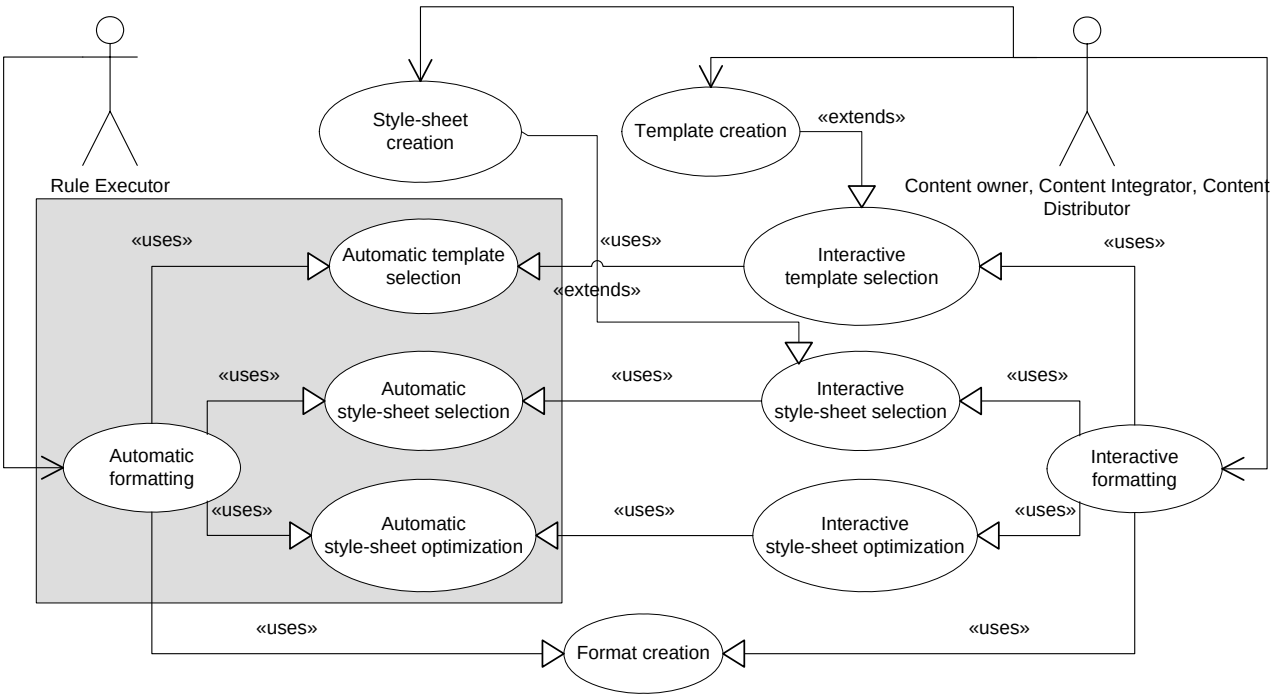
Scenario description:

1. **Start process**. The AXCP Rule Engine receives a running rule request coming from the AXMEDIS Workflow Manager, or from the AXMEDIS Publication & Programme, or from the internal scheduler that activates an AXCP rule.
2. **Rule execution request**. The scheduler sends the rule execution request to the rule executor with the corresponding rule (*Running rule*).
3. **AXMEDIS Objects selection request**. For each selection and/or query specified in the rule, the rule executor sends queries to the AXMEDIS Query Support to obtain references to AXMEDIS objects that match the request.
4. **AXOM usage**. An embedding object request with the relative object reference is sent to the AXOM to perform the inclusion.
5. **Physical Objects request**. The physical object is requested to the AXMEDIS Database by means of its reference.
6. **Adaptation request**. This request is performed via AXOM in order to perform a formatting paradigm or a set of customised formatting parameters. This phase could:
    a. Perform adaptation algorithm (change resolution, change dimension, time or spatial best fitting, etc…)
    b. Apply spatial and temporal constraints specified in the rule (i.e. graphical layout, temporisation, transitions effects, etc…)
    c. Perform synchronisation algorithm (audio and text audio and images, video and text, etc…)
    d. Convert the whole formatted object into a specific output format (i.e. MPEG4).
7. **External tools calling**. This request allows calling external functionalities available on external formatting tools. In this way some formatting operation can be delegated and performed in other formatting environment. These calls are properly coded in the formatting rule and refer to portion of code written by using for example the script language available on the external tool.
8. **Fingerprint request**. This request is performed via AXOM in order to apply the fingerprint to the formatted object according to the fingerprint parameters specified in the rule.

9. **Protection request**. This request is performed via AXOM. A protection request is sent to the Protection tool in order to apply protection to the new AXEMDIS formatted object.
10. **Storing AXMEDIS object**. The new formatted AXMEDIS Object is stored into the AXMEDIS Database.
11. **End process notification**. The End of the formatting process is notified to the AXMEDIS Workflow Manager.

2.3 Use Case:

The use case shows usage of high-level functionalities offered by Content Formattting Engine



Automatic formatting:

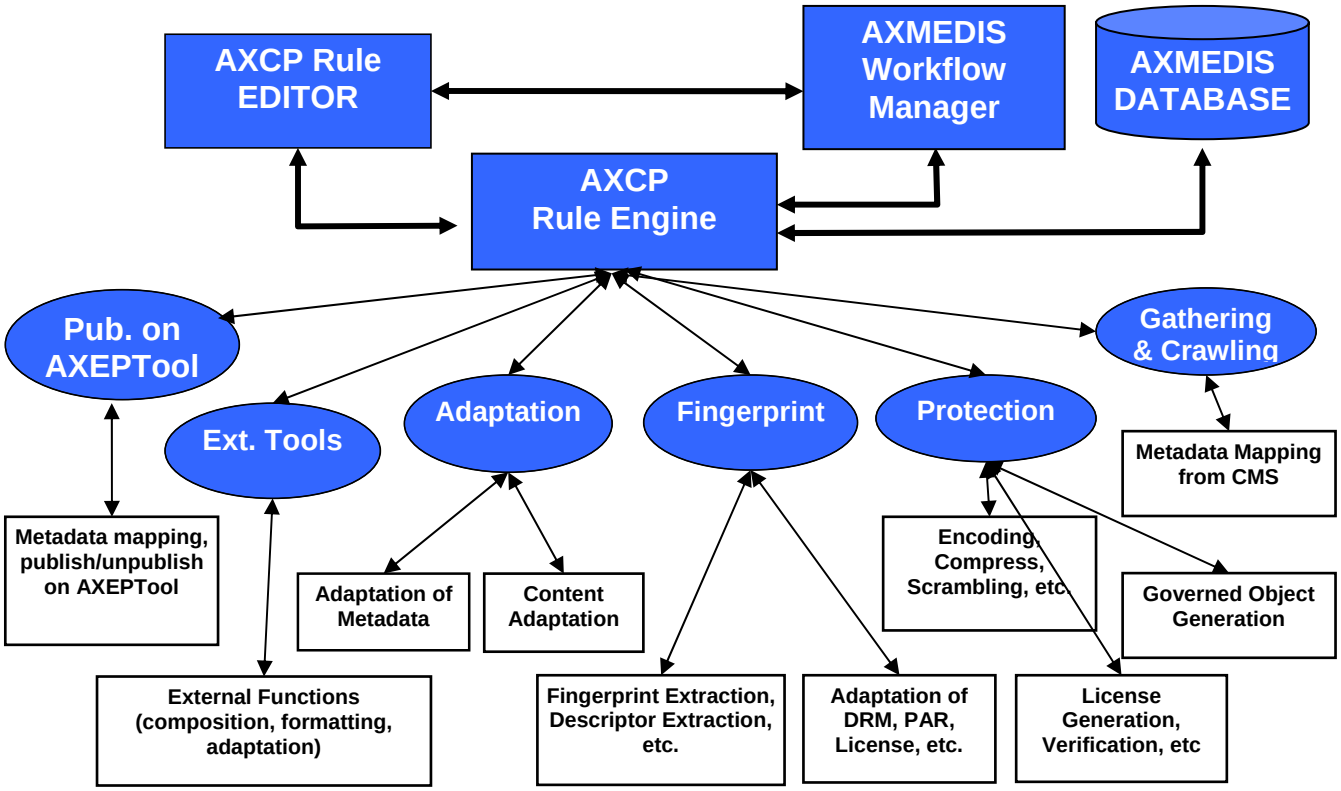
1. Automatic template selection
2. Automatic style-sheet selection
3. Automatic style-sheet optimization
4. Format creation

Interactive formatting:

1. Interactive template selection
   1.1. A new template may be created
2. Interactive style-sheet selection
   2.1. A new style-sheet may be created
3. Interactive style-sheet optimization
4. Format creation

3 AXMEDIS Content Processing Area General architecture and relationships among the modules produced

In this section, main actors involved in the AXMEDIS Content Processing Area are described. The picture, reported below, shows the unified solution and relationships among the AXMEDIS Workflow Manager, the AXCP Rule Editor and Engine plus the AXMEDIS Content Processing tools.



1. **Workflow Manager** – It performs the role of supervisor by monitoring and controlling the AXCP Rule engine and editor activity.
2. **AXCP Rules Editor –** It is an editor for writing protection, production and publication on AXEPTool rules. It is supported by a repository of rule rules and interacts with the AXCP Rule Engine.
3. **AXCP Rule Engine –** It performs the automatic production, protection, adaptation, gathering, P2P publication processes by means specific AXCP rules. It generates new composite, formatted, adapted, protected AXMEDIS objects, gathers raw contents, etc…. Such objects are successively stored in the AXMEDIS database or delivered via distribution channels or published by means the AXEPTool. The Engine is supported by following modules and tools:
    a. **Protection** – It provides functionalities and algorithms performing the protection of the AXMEDIS object via Protection Support
    b. **Fingerprint** – It provides functionalities and algorithms performing the Fingerprint estimation of a new AXMEDIS object. The fingerprint could be based on component's fingerprint or could be a new one.
    c. **Adaptation** – It provides functionalities and algorithms performing the Content adaptation for different distribution channels and format paradigm.
    d. **External tools** - Tools for using external formatting functionalities. It provides a set of plug-ins that allow using external tools (Macromedia suite, Adobe suite, etc…) and extending functionalities of the composition and formatting engine and AXMEDIS Editor.
    e. **Publication on AXEPTool** – It provides functionalities and algorithms performing metadata manipulation and mapping, publication of AXMEDIS object on the AXEPTool.

3.1 AXCP Content Processing area UML Decomposition

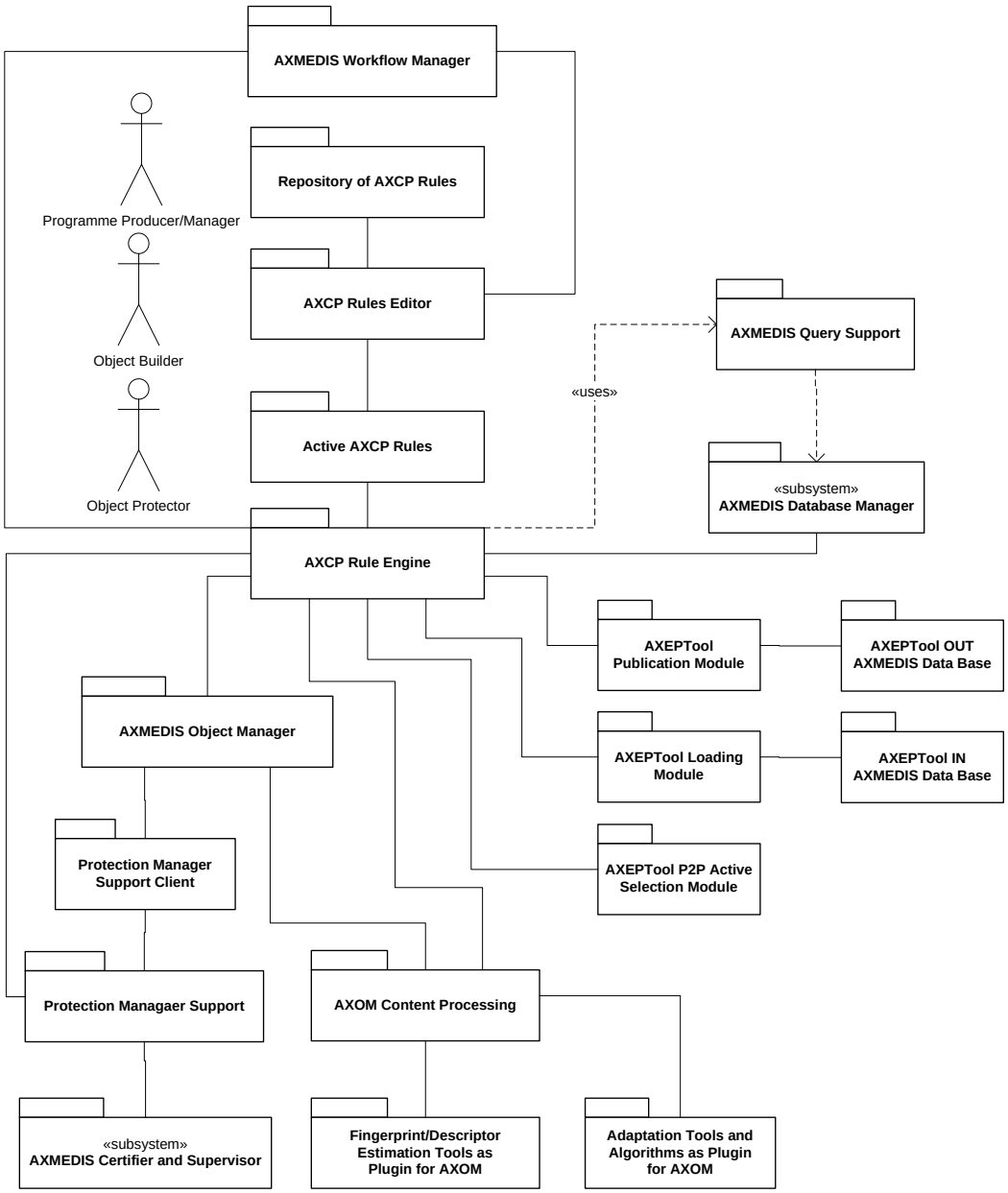
In this section the UML decomposition of the AXMEDIS Content Processing Area is described. It represents the unified solution based on the AXMEDIS Content Processing Rule Engine (**AXCP Rule Engine**) and that integrates the following engines:

- AXMEDIS Composition and Formatting Engine
- AXEPTool Loading Tool Engine
- AXEPTool Publication Tool Engine
- AXEPTool P2P Active Selection Engine
- The Protection Tool Engine

According to the UML diagram, the AXMEDIS Content Processing Area includes:

- **AXCP Rule Editor**: A graphic editor that allows writing and editing composition, formatting, adaptation, crawling, protection and publication on AXEPTool rules.
- **Repository of AXCP Rules**: It is a simple repository of rules; it is the file systems or a database. The rules are described by means an XML schema.
- **Active Rules**: They are rules that are scheduled to be run by the AXCP Rule Engine.
- **AXCP Rule Engine**: It consists in a scheduler of rules and a set of rule executors placed in a GRID environment.
- **AXMEDIS Database Manager** – It allows the engine to retrieve AXMEDIS objects involved in the execution of a rule.
- **AXMEDIS Query Support** – It allows the engine to sumbit queries to the AXMEDIS Database Manager
- **AXMEDIS Object Manager (AXOM)** – It allows managing AXMEDIS objects during the execution of a rule. It provides functions and methods for manipulating and managing resources and metadata,
- **AXOM Content Processing –** It is based on a Plugin Manager and the External Procedures Profile Manager. It allows extending the engine capabilities by providing the interface to Plug-ins such as:
    - **Adaptation Tools and Algorithms as Plugin for AXOM**: They are a collection of algorithms and tools that provides functions for content adaptation. The role of such component is to provided different methods to manipulate digital contents in order to satisfy several and different user profile.
    - **Fingerprint/Descriptor Estimation Tools as Plugin for AXOM -** They are a collection of algorithms and tools that provide functions for fingerprint/descriptors estimation from digital contents.
- **Protection Manager support –** It provides the support to manage the protection of AXMEDIS objects.
- **AXEPTool Publication Module** – It provides the support to manage the publication aspects in the AXEPTool
- **AXEPTool Loading Module** - It provides the support to manage the publication aspects in the AXEPTool
- **AXEPTool P2P Active Selection Module** - It provides the support to manage the publication aspects in the AXEPTool

AXCP Content Processing AREA



4 AXCP Rule Model

Module/Tool Profile AXCP Rule Model		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)		
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Mutlithread	
Language of Development	C++	
Platforms supported	MS Windows, Linux and Mac OS-X	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/source/rulemodel/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.axmedis.org/repos/Applications/ruleeditor/ https://cvs.axmedis.org/repos/Applications/ruleexecutor/	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	NO	
Usage of the AXMEDIS Error Manager (yes/no)	NO	
Major Problems not solved	-- --	
Major pending requirements	Management of scripts used as dependency. --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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++	XERCES 2.6.0	LGPL

4.1 General Description of the Module

The entire production process is driven by rules called AXMEDIS Content Processing rules (AXCP Rule). They allow an automatic and customizable process that responds to the distribution and end user needs.
A rule have to:

- describe what resources are involved in the processing (i.e. extracting digital resources from the AXMEDIS database by means of queries built on metadata and licensing information or from a composite AXMEDIS object);
- describe distribution channel properties, user device features, user profile, etc…
- describe the final output using a specific integration format (MPEG-4, SMIL,…) or using DIP capabilities provided by MPEG-21 objects
- describe how to combine different digital resources and create relationships in terms of:
    - spatial relationships (for graphic layout, resource adaptation, …)
    - time relationships (for synchronisation, transitions effect, fitting (shrinking or stretching, cutting,)…);
- describe how to manage and combine DRM rules for the new formatted resource;
- describe operations or actions that have to be performed during the formatting process, for example:
    - which formatting algorithms have to be used (synchronisation, image scaling, resolution scaling, format conversion, etc…)
    - which external functionalities (by dynamic call to services provided by external tools) have to be used
    - Fingerprint estimation and application for the new composite item
    - Object ID assignment for the new composite item.
- describe how to protect resources

In general, a rule could be formalised as a function in the following way:

$$R = f(S1,S2,..,Sn,P1,\ldots,Pm)$$

Where:

- S_i – It defines a selection. It is a sequence of query to be sent to the Query Support for AXMEDIS objects retrieval or references to digital resource embedded into an composite AXEMDIS object;
- P_i – It is a parameter (basic type as integer, string, Boolean);
- *f* is the identifier of rule (name of rule or other);
- *R* is the resultant of rule application. It will be a new AXMEDIS object, or a metadata manipulation, the protection of an AXMEDIS object, etc…

In this section the structure of a rule is described. A rule is constituted of three main sections:

- **Header –** General metadata about the AXCP rule
- **Schedule –** Temporal metadata that describes conditions for firing the AXCP rule
- **Definition –** The definition of the AXCP rule

Header

This section contains metadata related to general information associated with a rule. It is constituted of::

Header				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Rule Name	String	e.g. “Audio Collection”	It defines the name of the rule	
AXRID	String		It defines the AXMEDIS Rule ID	
Rule Version	String	e.g. “1.0”	It defines the version of the rule	
Rule Type	String	AXCP Rule, AXPnP Rule	It defines the type of rule	AXCP rules identifies all rules related to the Content Processing Area, whereas the AXPnP rules are the rule of the P&P area
Software Name	String	“Axmedis Rule Editor”	It specifies the name of software used	
Version of software	String	e.g. “2.0”	It defines the version of software used.	
Date of production	Date	dd.mm.yy	It defines when the rule has been created	
Time of production	Time	hh.mm.ss am/pm	It defines at what time the rule has been created	
Author	String	e.g. “John Brown”	It defines the name of author who has created the rule	
Affiliation	String	e.g. “DSI”	It defines the name of Affiliation	
URL	String	e.g. “http://www....”	It defines the Internet address of the Affiliation	
Comment	String		It allows describing what the rule does	
Last_Modification	Date		Who is last modified	
Terminal_ID	String		The Id of the terminal used to write the rule.	
Cost	Enum		Estimation of Cost	

Work_Item_ID	String		External reference, for instance the commitment	

Schedule
This section contains the sequence of metadata for programming the activation of a rule:

Schedule				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Run	Section		It defines a subsection of metadata that describe information needed for scheduling the execution of the rule.	
Status	String	"Active", "Inactive"	It defines if a rule: • is active and can be executed • is inactive	The list of status identifiers could be extended if it is necessary.

Run				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Start Date	Date	dd.mm.yy	It defines when the rule has to be executed by the engine in terms of day, month and year.	
Start Time	Time	hh.mm.ss am/pm	It defines when the rule has to be executed by the engine in term of time clock.	
Periodicity	String	"Monthly", "Daily", "Weekly", etc…	It defines if a rule has to be executed periodically	Optional
Expiration date	Date	dd.mm.yy	To stop the periodicity	Optional
Expiration time	Time	hh.mm.ss am/pm	To stop the periodicity	Optional

Definition
This section include the *AXCP Rule* section containing the procedural description of the rule.

AXCP Rule				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Arguments	Section		It includes the set of *selections* and *parameters* that rule has in input.	
Dependencies	Section		It includes the list of DLL/functions, Plug In used by the script preconditions	This is information could be used to check the feasibility of rule before running it
Rule Body	Section		It includes the JavaScript code that defines the Rule	
Files Complexity	TBD	TBD	It will specify the complexity of the rule in terms of computational, file transfer parameters, estimated amount of disk space required by the digital resources involved in the rule and other parameters	This will be better defined during the project life

The Arguments subsection contains the list of Selections and Parameters that will be used by the rule.

For each *Selection* see the "XML Selection Schema" in Part E.

Each *Parameter* is defined as following:

Parameter				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Name	String		It specifies the name of the parameter	
Type	String	Integer, String, Float, Double, etc…	It specifies the type of parameter	
Value	string		It specifies the value that the parameter assumes	

The Preconditions subsection contains information about the AXMEDIS Editor Plug In that could be required by the Rule Body. This mechanism is similar to the import directive in JAVA language.

Preconditions				
Data	**Type**	**Values/Format**	**Description**	**Issues**
Plug_In_Name	String	e.g.: “Adaptation"	It provides the name of the AXMEDIS Editor Plug In used by the script. This information has to be matched with that provided by the DLL from its profile.	
Version	String	e.g.: “2.0”	Version of the Plug In. This information has to be matched with that provided by the DLL from its profile.	

The Rule Body section provides two possible ways to refer the adopted script:

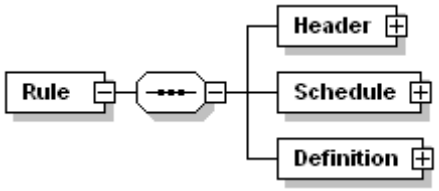
Rule Body				
Data	**Type**	**Values/Format**	**Description**	**Issues**
JS_Script	String	e.g. “Script Name”	It could be used to embed the whole script (JavaScript code) inside the XML rule format.	Choice
Path	URL		It could be used to specifies reference to a “.js” file that contains the source script of the current rule (JavaScript code).	Optional

4.2 AXCP Rule XMLSchema

The following XML schema refers to the “Rule_Axmedis.xsd” file.

element **Rule**

diagram



children **Header** **Schedule** **Definition**

source

```
<xs:element name="Rule">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Header">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Rule_Name" type="xs:string"/>
```

```
      <xs:element name="AXRID" type="xs:string"/>
      <xs:element name="Rule_Version" type="xs:string"/>
      <xs:element name="Rule_Type">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="AXCP"/>
         <xs:enumeration value="AXPnP"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:element>
      <xs:element name="Software_Name" type="xs:string"/>
      <xs:element name="Version_of_software" type="xs:string"/>
      <xs:element name="Date_of_production" type="xs:date"/>
      <xs:element name="Author" type="xs:string"/>
      <xs:element name="Affiliation" type="xs:string"/>
      <xs:element name="URL" type="xs:anyURI"/>
      <xs:element name="Comment" type="xs:string"/>
      <xs:element name="Last_Modifications" type="xs:date"/>
      <xs:element name="Terminal_ID" type="xs:string"/>
      <xs:element name="Cost" type="xs:string"/>
      <xs:element name="Work_Item_ID" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="Schedule">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Run">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Time" type="xs:time"/>
         <xs:element name="Periodicity" minOccurs="0">
          <xs:complexType>
           <xs:simpleContent>
            <xs:extension base="xs:integer">
             <xs:attribute name="Unit" type="periodunit"/>
            </xs:extension>
           </xs:simpleContent>
          </xs:complexType>
         </xs:element>
         <xs:element name="Expiration_Date" type="xs:date" minOccurs="0"/>
         <xs:element name="Expiration_Time" type="xs:time" minOccurs="0"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element name="Status">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="Active"/>
         <xs:enumeration value="Inactive"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="Definition">
    <xs:complexType>
     <xs:choice minOccurs="0">
      <xs:element name="AXCP_Rule">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Arguments">
          <xs:complexType>
           <xs:choice maxOccurs="unbounded">
            <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
             <xs:complexType>
              <xs:simpleContent>
               <xs:extension base="xs:string">
                <xs:attribute name="Name" type="xs:string" use="required"/>
                <xs:attribute name="Type" type="xs:string" use="required"/>
               </xs:extension>
              </xs:simpleContent>
             </xs:complexType>
            </xs:element>
```

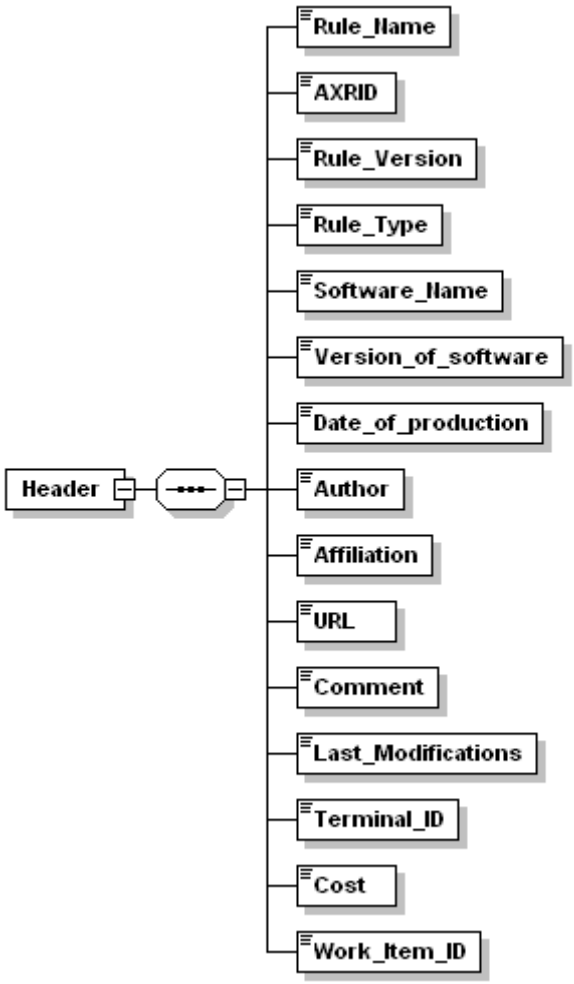
```
                    <xs:element ref="selection" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:choice>
                 </xs:complexType>
                </xs:element>
                <xs:element name="Rule_Body">
                 <xs:complexType>
                  <xs:choice>
                   <xs:element name="JS_Script" type="xs:string"/>
                   <xs:element name="Path" type="xs:anyURI" minOccurs="0"/>
                  </xs:choice>
                 </xs:complexType>
                </xs:element>
                <xs:element name="Dependencies" minOccurs="0">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                     <xs:sequence>
                      <xs:element name="Plug_In_name" type="xs:string"/>
                      <xs:element name="Version" type="xs:string"/>
                     </xs:sequence>
                    </xs:complexType>
                   </xs:element>
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
              </xs:complexType>
             </xs:element>
             <xs:element name="PnP_Rule"/>
            </xs:choice>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
```

description **A rule is constituted of three main sections:**

- Header **– General metadata about the AXCP rule**
- Schedule **– Temporal metadata that describes conditions for firing the AXCP rule**
- Definition **– The definition of the AXCP rule**

element **Rule/Header**

diagram



children Rule_Name AXRID Rule_Version Rule_Type Software_Name Version_of_software Date_of_production Author Affiliation URL Comment Last_Modifications Terminal_ID Cost Work_Item_ID

source

```
<xs:element name="Header">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Rule_Name" type="xs:string"/>
   <xs:element name="AXRID" type="xs:string"/>
   <xs:element name="Rule_Version" type="xs:string"/>
   <xs:element name="Rule_Type">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="AXCP"/>
      <xs:enumeration value="AXPnP"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="Software_Name" type="xs:string"/>
   <xs:element name="Version_of_software" type="xs:string"/>
   <xs:element name="Date_of_production" type="xs:date"/>
   <xs:element name="Author" type="xs:string"/>
   <xs:element name="Affiliation" type="xs:string"/>
   <xs:element name="URL" type="xs:anyURI"/>
   <xs:element name="Comment" type="xs:string"/>
   <xs:element name="Last_Modifications" type="xs:date"/>
   <xs:element name="Terminal_ID" type="xs:string"/>
   <xs:element name="Cost" type="xs:string"/>
   <xs:element name="Work_Item_ID" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
```

```
</xs:element>
```

description This section contains metadata related to general information associated with a rule

element **Rule/Header/Rule_Name**

diagram 

type **xs:string**

source
```
<xs:element name="Rule_Name" type="xs:string"/>
```

description It defines the name of the rule, e.g. “Audio Collection”

element **Rule/Header/AXRID**

diagram 

type **xs:string**

source
```
<xs:element name="AXRID" type="xs:string"/>
```

description It defines the AXMEDIS Rule ID

element **Rule/Header/Rule_Version**

diagram 

type **xs:string**

source
```
<xs:element name="Rule_Version" type="xs:string"/>
```

description It defines the version of the rule, e.g. “1.0”

element **Rule/Header/Rule_Type**

diagram 

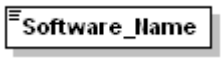
type restriction of **xs:string**

facets
enumeration AXCP
enumeration AXPnP

source
```
<xs:element name="Rule_Type">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="AXCP"/>
      <xs:enumeration value="AXPnP"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

description It defines the type of rule, AXCP rules identifies all rules related to the Content Processing Area, whereas the AXPnP rules are the rule of the P&P area

element **Rule/Header/Software_Name**

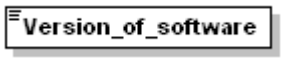
diagram 

type **xs:string**

source
```
<xs:element name="Software_Name" type="xs:string"/>
```

description It specifies the name of software used, e.g. “Axmedis Rule Editor”

element **Rule/Header/Version_of_software**

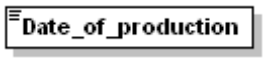
diagram 

type **xs:string**

source <xs:element name="Version_of_software" type="xs:string"/>

description It defines the version of software used., e.g. “2.0”

element **Rule/Header/Date_of_production**

diagram 

type **xs:date**

source <xs:element name="Date_of_production" type="xs:date"/>

description It defines when the rule has been created

Note This item now embeds the item Time_of_Production defined in the DE3-1-2C-AXFW_Spec-(content-production)-Part-C document.

:

element **Rule/Header/Author**

diagram 

type **xs:string**

source <xs:element name="Author" type="xs:string"/>

description It defines the name of author who has created the rule

element **Rule/Header/Affiliation**

diagram 

type **xs:string**

source <xs:element name="Affiliation" type="xs:string"/>

description It defines the name of Affiliation

element **Rule/Header/URL**

diagram 

type **xs:anyURI**

source <xs:element name="URL" type="xs:anyURI"/>

description It defines the Internet address/URL of the Affiliation

element **Rule/Header/Comment**

diagram 

type **xs:string**

source <xs:element name="Comment" type="xs:string"/>

description It allows describing what the rule does

element **Rule/Header/Last_Modifications**

diagram 

type **xs:date**

source <xs:element name="Last_Modifications" type="xs:date"/>

description It is used to report the last modified date

element **Rule/Header/Terminal_ID**

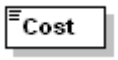
diagram 

type **xs:string**

source <xs:element name="Terminal_ID" type="xs:string"/>

description The Id of the terminal used to write the rule.

element **Rule/Header/Cost**

diagram 

type **xs:string**

source <xs:element name="Cost" type="xs:string"/>

description Estimation of Cost

element **Rule/Header/Work_Item_ID**

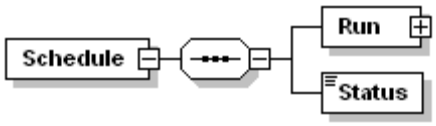
diagram 

type **xs:string**

source <xs:element name="Work_Item_ID" type="xs:string"/>

description External reference, for instance the commitment

element **Rule/Schedule**

diagram 

children **Run Status**

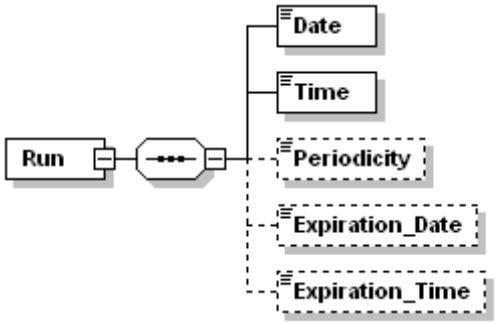
source
```
<xs:element name="Schedule">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Run">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Date" type="xs:date"/>
            <xs:element name="Time" type="xs:time"/>
            <xs:element name="Periodicity" minOccurs="0">
              <xs:complexType>
```

```
            <xs:simpleContent>
              <xs:extension base="xs:integer">
                <xs:attribute name="Unit" type="periodunit"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Expiration_Date" type="xs:date" minOccurs="0"/>
        <xs:element name="Expiration_Time" type="xs:time" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Status">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Active"/>
        <xs:enumeration value="Inactive"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
</xs:element>
```

description This section contains the sequence of metadata for programming the activation of a rule

element **Rule/Schedule/Run**

diagram



children **Date Time Periodicity Expiration_Date Expiration_Time**

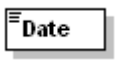
source

```
<xs:element name="Run">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Date" type="xs:date"/>
   <xs:element name="Time" type="xs:time"/>
   <xs:element name="Periodicity" minOccurs="0">
    <xs:complexType>
     <xs:simpleContent>
      <xs:extension base="xs:integer">
       <xs:attribute name="Unit" type="periodunit"/>
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="Expiration_Date" type="xs:date" minOccurs="0"/>
   <xs:element name="Expiration_Time" type="xs:time" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

description It defines a subsection of metadata that describes information needed for scheduling the execution of the rule.

element **Rule/Schedule/Run/Date**

diagram

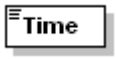


type **xs:date**

source <xs:element name="Date" type="xs:date"/>

description It defines when the rule has to be executed by the engine in terms of day, month and year.

element **Rule/Schedule/Run/Time**

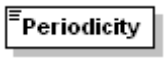
diagram 

type **xs:time**

source <xs:element name="Time" type="xs:time"/>

description It defines when the rule has to be executed by the engine in term of time clock.

element **Rule/Schedule/Run/Periodicity**

diagram 

type extension of **xs:integer**

attributes

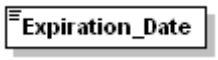
Name	Type	Use	Default	Fixed	Annotation
Unit	periodunit				

source

```
<xs:element name="Periodicity" minOccurs="0">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:integer">
    <xs:attribute name="Unit" type="periodunit"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

description It defines if a rule has to be executed periodically, every “Unit” “periodunit” e.g. “2” “Week”

element **Rule/Schedule/Run/Expiration_Date**

diagram 

type **xs:date**

source <xs:element name="Expiration_Date" type="xs:date" minOccurs="0"/>

description The date to stop the periodicity

element **Rule/Schedule/Run/Expiration_Time**

diagram 

type **xs:time**

source <xs:element name="Expiration_Time" type="xs:time" minOccurs="0"/>

description The time to stop the periodicity

element **Rule/Schedule/Status**

diagram 

type restriction of **xs:string**

facets

enumeration	Active
enumeration	Inactive

source

```
<xs:element name="Status">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="Active"/>
   <xs:enumeration value="Inactive"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

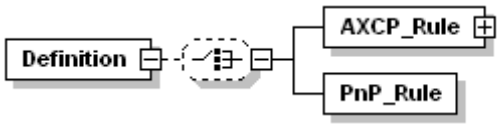
description

It defines if a rule is:

- "active" and can be executed
- "inactive"

element **Rule/Definition**

diagram



children **AXCP_Rule PnP_Rule**

source

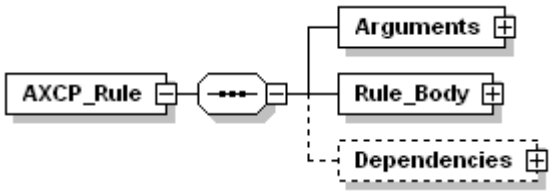
```
<xs:element name="Definition">
 <xs:complexType>
  <xs:choice minOccurs="0">
   <xs:element name="AXCP_Rule">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Arguments">
       <xs:complexType>
        <xs:choice maxOccurs="unbounded">
         <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
           <xs:simpleContent>
            <xs:extension base="xs:string">
             <xs:attribute name="Name" type="xs:string" use="required"/>
             <xs:attribute name="Type" type="xs:string" use="required"/>
            </xs:extension>
           </xs:simpleContent>
          </xs:complexType>
         </xs:element>
         <xs:element ref="selection" minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
       </xs:complexType>
      </xs:element>
      <xs:element name="Rule_Body">
       <xs:complexType>
        <xs:choice>
         <xs:element name="JS_Script" type="xs:string"/>
         <xs:element name="Path" type="xs:anyURI" minOccurs="0"/>
        </xs:choice>
       </xs:complexType>
      </xs:element>
      <xs:element name="Dependencies" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="Plug_In_name" type="xs:string"/>
            <xs:element name="Version" type="xs:string"/>
           </xs:sequence>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="PnP_Rule"/>
  </xs:choice>
```

```
  </xs:complexType>
</xs:element>
```

description This section includes the section containing the procedural description of an AXCP or a PnP rule

element **Rule/Definition/AXCP_Rule**

diagram



children **Arguments Rule_Body Dependencies**

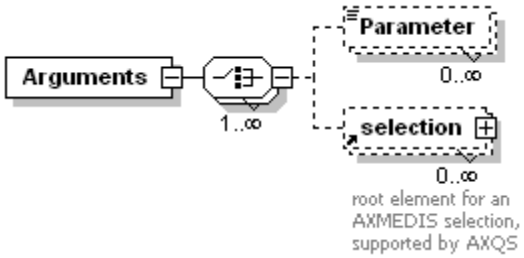
source

```
<xs:element name="AXCP_Rule">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Arguments">
    <xs:complexType>
     <xs:choice maxOccurs="unbounded">
      <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:simpleContent>
         <xs:extension base="xs:string">
          <xs:attribute name="Name" type="xs:string" use="required"/>
          <xs:attribute name="Type" type="xs:string" use="required"/>
         </xs:extension>
        </xs:simpleContent>
       </xs:complexType>
      </xs:element>
      <xs:element ref="selection" minOccurs="0" maxOccurs="unbounded"/>
     </xs:choice>
    </xs:complexType>
   </xs:element>
   <xs:element name="Rule_Body">
    <xs:complexType>
     <xs:choice>
      <xs:element name="JS_Script" type="xs:string"/>
      <xs:element name="Path" type="xs:anyURI" minOccurs="0"/>
     </xs:choice>
    </xs:complexType>
   </xs:element>
   <xs:element name="Dependencies" minOccurs="0">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Plug_In_name" type="xs:string"/>
         <xs:element name="Version" type="xs:string"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

description This section include the AXCP Rule section containing the procedural description of the rule

element **Rule/Definition/AXCP_Rule/Arguments**

diagram



children **Parameter** **selection**

source

```
<xs:element name="Arguments">
 <xs:complexType>
  <xs:choice maxOccurs="unbounded">
   <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:simpleContent>
      <xs:extension base="xs:string">
       <xs:attribute name="Name" type="xs:string" use="required"/>
       <xs:attribute name="Type" type="xs:string" use="required"/>
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
   <xs:element ref="selection" minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

description It includes the set of selections and parameters that rule has in input.

element **Rule/Definition/AXCP_Rule/Arguments/Parameter**

diagram

type extension of **xs:string**

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xs:string	required			
Type	xs:string	required			

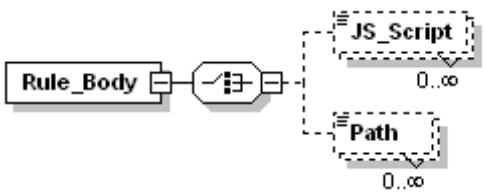
source

```
<xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="Type" type="xs:string" use="required"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

description It defines a parameter in input to the rule by providing the name, the type and the actual value

element **Rule/Definition/AXCP_Rule/Rule_Body**

diagram



children **JS_Script** **Path**

source
```
<xs:element name="Rule_Body">
 <xs:complexType>
  <xs:choice>
   <xs:element name="JS_Script" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:simpleContent>
      <xs:extension base="xs:string">
       <xs:attribute name="name" type="xs:string"/>
       <xs:attribute name="main" type="xs:boolean"/>
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="Path" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:simpleContent>
      <xs:extension base="xs:anyURI">
       <xs:attribute name="main" type="xs:boolean"/>
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

element **Rule/Definition/AXCP_Rule/Rule_Body/JS_Script**

diagram 

type extension of xs:string

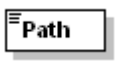
attributes

Name	Type	Use	Default	Fixed	Annotation
name	xs:string				
main	xs:boolean				

source
```
<xs:element name="JS_Script" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="main" type="xs:boolean"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

description It is used to embed the whole script (JavaScript code) inside the XML rule format.

element **Rule/Definition/AXCP_Rule/Rule_Body/Path**

diagram 

type extension of xs:anyURI

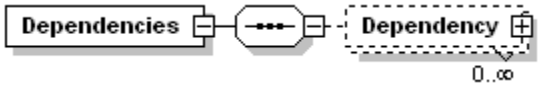
attributes

Name	Type	Use	Default	Fixed	Annotation
main	xs:boolean				

source
```
<xs:element name="Path" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:anyURI">
    <xs:attribute name="main" type="xs:boolean"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

description It is used to specify a reference to a ".js" file that contains the source script code of the current rule (JavaScript code).

element **Rule/Definition/AXCP_Rule/Dependencies**

diagram 

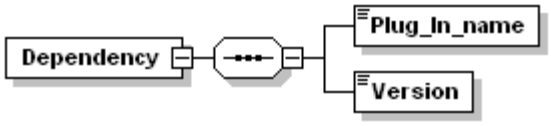
children **Dependency**

source
```
<xs:element name="Dependencies" minOccurs="0">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Plug_In_name" type="xs:string"/>
      <xs:element name="Version" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

description It contains a list of possible dependencies

Note This item replaces the Precondition item defined in the schema reported in DE3-1-2C-AXFW_Spec-(content-production)-Part-C document.

element **Rule/Definition/AXCP_Rule/Dependencies/Dependency**

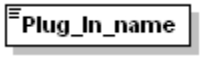
diagram 

children **Plug_In_name** **Version**

source
```
<xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Plug_In_name" type="xs:string"/>
   <xs:element name="Version" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

description It contains information about the AXMEDIS Editor Plug In that could be required by the Rule Body. This mechanism is similar to the import directive in JAVA language.

element **Rule/Definition/AXCP_Rule/Dependencies/Dependency/Plug_In_name**

diagram 

type **xs:string**

source
```
<xs:element name="Plug_In_name" type="xs:string"/>
```

description It provides the name of the AXMEDIS Editor Plug In used by the script. This information has to be matched with that provided by the DLL from its profile.

element **Rule/Definition/AXCP_Rule/Dependencies/Dependency/Version**

diagram 

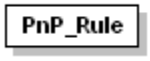
type **xs:string**

source
```
<xs:element name="Version" type="xs:string"/>
```

description Version of the Plug In. This information has to be matched with that provided by the DLL from its profile.

element **Rule/Definition/PnP_Rule**

diagram



source
```
<xs:element name="PnP_Rule"/>
```

simpleType **periodunit**

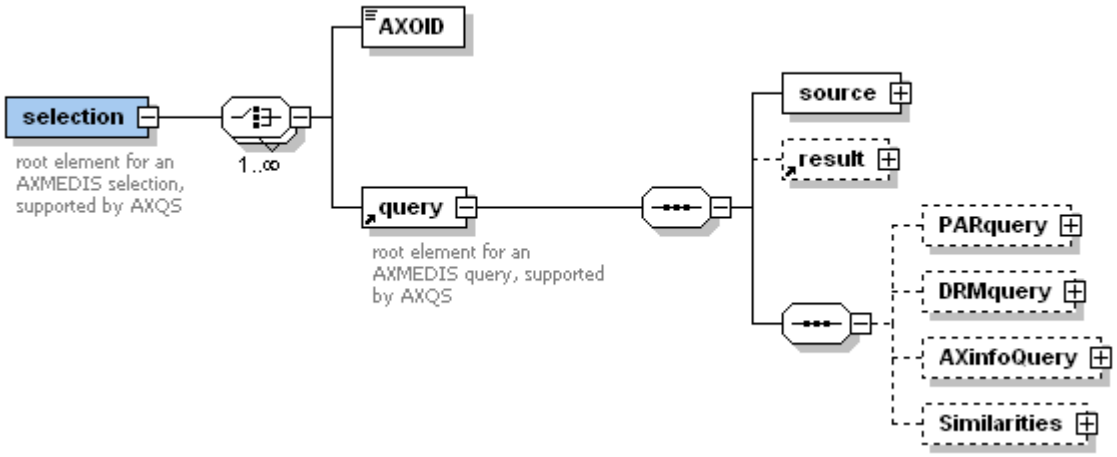
type restriction of **xs:string**

used by attribute **Rule/Schedule/Run/Periodicity/@Unit**

facets
enumeration Day
enumeration Month
enumeration Week
enumeration Year

source
```
<xs:simpleType name="periodunit">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Day"/>
    <xs:enumeration value="Month"/>
    <xs:enumeration value="Week"/>
    <xs:enumeration value="Year"/>
  </xs:restriction>
</xs:simpleType>
```

Selection XML schema is defined as following:



The file associated with the rule is an XML file whose name will be generated as following:

< Rule_Filename > = <Rule_Name> + <Rule_Version> + AXRID + ".xml "

4.3 Examples of usage

The following example shows the XML structure for a generic rule:

```
<Rule>
        <Header>
                <Rule_Name> Audio Collection </Rule_Name>
                <AXRID> 0010001 </AXRID>
                <Rule_Version> 1.0 </Rule_Version>
```

```
                <Rule_Type> Compositional </Rule_Type>
                <Software_Name> Axmedis Rule Editor </Software_Name>
                <Version_of_software> 2.0 </Version_of_software>
                <Date_of_production> 24.12.2004 </Date_of_production>
                <Time_of_production> 12:00 am </Time_of_production>
                <Author> John Brown </Author>
                <Affiliation> DSI </Affiliation>
                <URL> http://www.dsi.unifi.it </URL>
                <Comment> This rule embeds only audio file </Comment>
                <Last_Modifications> 25.12.2004 </Last_Modifications>
                <Terminal_ID> </Terminal_ID>
                <Cost> What is cost? </Cost>
                <Work_Item_ID> What? </Work_Item_ID>
        </Header>
        <Schedule>
                <Run>
                        <Date> 24.12.2004 <Date>
                        <Time> 12:00 pm </Time>
                        <Periodicity> Weekly </Periodicity>
                        <Expiration_Date> 01.01.2005 </Expiration_Date>
                        <Expiration_Time> 12:00 pm </Expiration_Time>
                 </Run>
                <Status> Active </Status>
        </Schedule>
        <Definition>
                <AXCP_Rule>
                        <Arguments>
                                <selection name="TEST" timestamp="2005-01-20T18:20:46.275+01:00">
                                <AXOID>3y7932469236</AXOID>
                                <AXOID>824375832741723</AXOID>
                                <query>
                                        <source>
                                                <location>CRAWLER</location>
                                        </source>
                                        <AXinfoQuery>
                                                <querycondition>
                                                        <nesting>
                                                                <test>
                                                                        <field>AUTHOR</field>
                                                                        <operator>STARTWITH</operator>
                                                                        <value>MOZ</value>
                                                                </test>
                                                        </nesting>
                                                </querycondition>
                                        </AXinfoQuery>
                                </query>
                                </selection>
                        <Parameter name="count" type = "integer" > 20 </ Parameter >
                        ……
                        <Parameter>  ……</ Parameter >
                        </ Arguments>
                        <Preconditions>
                        <Plug_In_Name> Adaptation </Plug_In_Name >
                        <Version> 2.0 </Version>
                        </Preconditions>
                        <Preconditions>
                        …….
                        </Preconditions>
                        <Rule_Body>
                            …
                        </Rule_Body>
                </AXCP_Rule>
        </Definition>
</Rule>
```

4.4 AXCP Rule Model

According to the XML schema, the object oriented model of the rule is described by the class diagram reported in the Figure A and B. Actually, the current status of the model is close to the final version. Implementation was realised in C++ MSVC7 and supported by the XERCES 2.6.0 libraries.

AxDOMImplementation Class

The *AxDOMImplementation* class specialises the *ErrorHandler* interface class of the XERCES library ver. 2.6.0. by redefining the pure virtual methods:

- virtual void **warning** (const **SAXParseException** &exc)=0
  Receive notification of a warning. http://xml.apache.org/xerces-c/apiDocs/classErrorHandler.html - z819_0#z819_0
- virtual void **error** (const **SAXParseException** &exc)=0

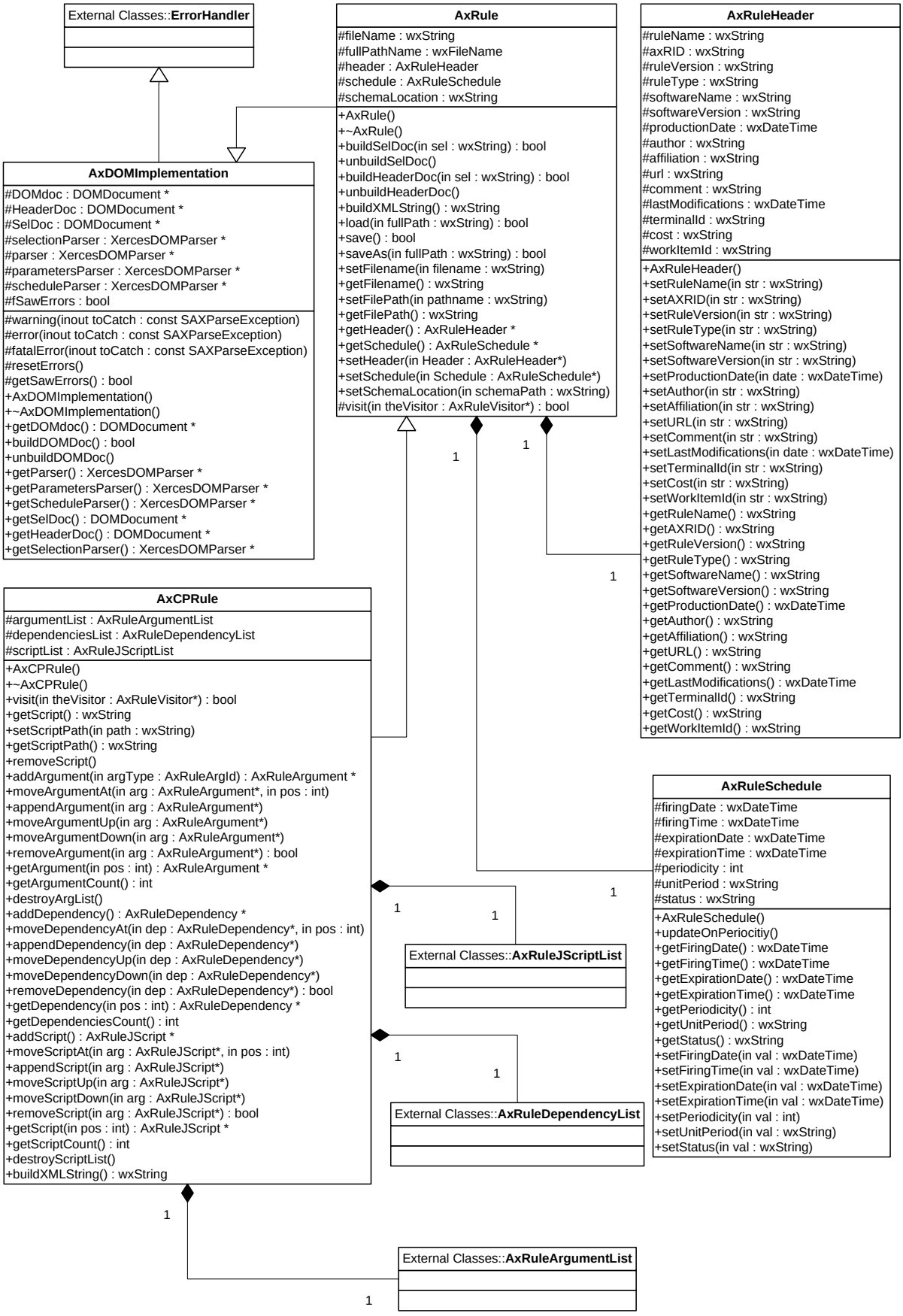
Receive notification of a recoverable error. http://xml.apache.org/xerces-c/apiDocs/classErrorHandler.html - z819_1#z819_1

- virtual void **fatalError** (const **SAXParseException** &exc)=0
  Receive notification of a non-recoverable error. http://xml.apache.org/xerces-c/apiDocs/classErrorHandler.html - z819_2#z819_2
- virtual void **resetErrors** ()=0
  Reset the Error handler object on its reuse.

Such methods are called if the validation of the XML Rule file fails during the parsing. The *AxDOMImplementation* class does not consider warnings and puts the *fSawErrors* attribute at *true* when an error occurs. The *getSawErrors()* method returns the *fSawErrors* value and then allows evaluating the result of validation. The class provides a pointer to the DOM tree (*DOMDoc*) of the all XML rule and is fixed by the *buildDOMDoc()* method. Such method was defined as virtual demanding its implementation to the *AxRule* class. It is equipped with a *XercesDOMParser* (*parser*) for loading the whole rule and *XercesDOMParser* (*parser*) for managing the XML string of a *Selection* in independent manner when saving the rule.

AxRuleHeader Class
The rule header is the class for storing the AxRule header information with access methods. It manages all properties of the *Header* section according to the XML schema. It provides getter and setter methods to access attributes.

AxRuleSchedule Class
The rule schedule is the class for storing the AxRule schedule information with access methods. It manages all properties of the *Schedule* section according to the XML schema. It provides getter and setter methods to access attributes.



Rule Model Class Diagram (A)

AxRule Class
The *AxRule* is the main class for a rule. It specialises the *AxDOMImplementation* and redefines the *buildDOMDoc()* method for building the DOM tree during the load and save phases. It encapsulates the *AxRuleHeader* and the *AxRuleSchedule* class. In this way, the *AxRule* class is the common class for the AXCP and AXPnP rules. Each of them has to define the *Definition* section according to Rule XML Schema in order to specialise the AXRule. The AxRule class provides the *buildXMLString()* to build a string reporting the XML code of the rule, the *load()*, *save()* and *saveAs()* methods. The loading and savig methods instantiate an *AxRuleVisitor* object that is passed as argument to the *visit()* method. The *AxRuleVisitor* object will be an *AxRuleLoader* when loading and an *AxRuleSaver* when saving; in both case the polymorphic *visit()* method is called

AxCPRule Class
It specialises the AxRule class and inherits all methods. It adds methods and attributes to manage the *Definition* section of the XML schema. *AxRuleSScriptList*, *AxRuleArgumentList* and *AxRuleDependencyList* contain respectively *AxRuleJSScript*, *AxRuleArgument* and *AxRuleDependecy* object, they model scripts, arguments and dependencies of the rule. The class provides also direct access methods to the internal list (getter and setter, item counting, access and removal).

AxRuleJScriptList
This class manages the list of javascript objects, It provides methods for accessing, reading, adding, removing a *AxRuleJScript* object.

AxRuleJScript
The AxRuleJSScript models a javascript object. The class provides also the *scriptName* of script (e.g. filename). The javascript source code is stored in the *javaScriptCode* string attribute. Optionally, a file with the javascript code could be referred by means the *path* string attribute. The *included* Boolean attribute stands for a status of the script: when it is *true* the script is included by reference using *path* otherwise is embedded in the XML of rule. The *client* void pointer stands for a reference to a data structure such as client data in the GUI.

AxRuleArgumentList
This class manages the list of arguments, It provides methods for accessing, reading, adding, removing a *AxRuleArgument* object.

AxRuleArgument
The AxRuleArgument models a generic argument of rule. The *argId* attributes specifies if the argument is a parameter (*axID_PARAMETER*) or a selection (*axID_SELECTION*). The class provides also the *name* attribute.

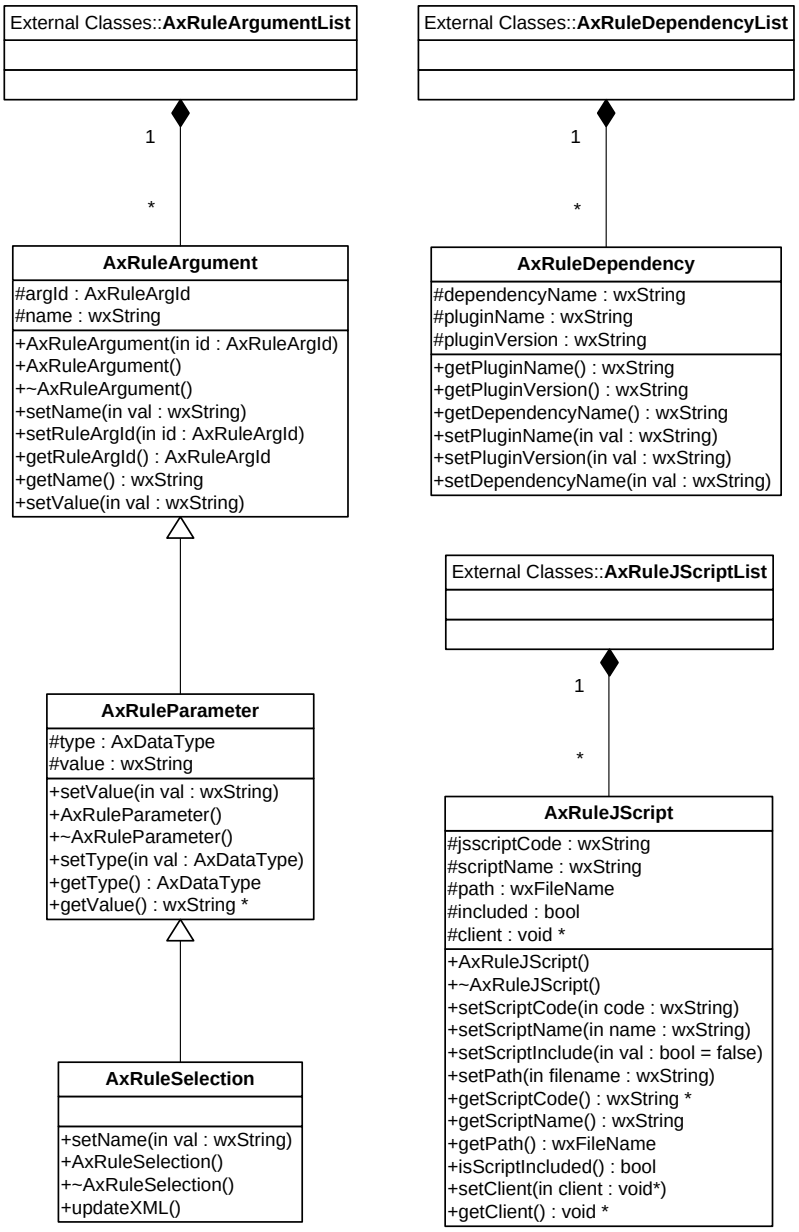
AxRuleParameter
It specialises the *AxRuleArgument* by defining the *type* and the *value* of the parameter. The *value* attribute is stored as string,whereas the *type* is associated with the enumerate values (*axINTEGER =0*, *axREAL* ,*axSTRING*, *axXMLSTRING*, *axDATE*, *axTIME*, *axBOOLEAN*, *axURL*, *axNULL*).

AxRuleSelection
It is a particular type of parameter. A selection is stored as a full XML string according to the Selection schema. The *AxRuleSelection* class specialises the *AxRuleParameter* by adding a method (*updateXML()*) to manage the change of the selection name directly on the XML string when calling the *setName()* method.

AxRuleDependencyList
This class manages the list of dependecies, It provides methods for accessing, reading, adding, removing a *AxRuleDependency* object.

AxRuleDependency

This class manages the dependency of the javascript code to a specific Axmedis plugin. It provides getter and setter methods to access *dependencyName*, *pluginName* and *pluginVersion* attributes.



Rule Model Class Diagram (B)

4.5 AXRule Loader and Saver Modules (DSI)

To manage the repository of rules it is necessary to load and save a rule in/from the system. The Rule Load and Save functionalities was designed and implemented by means of the following classes:

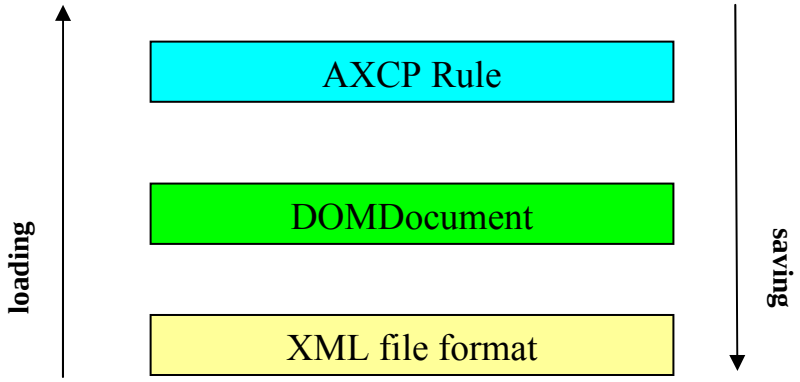
AXRule Loader - It is the module for loading an XML representation of the rule in the AXCP Rule Editor and AXCP Rule Engine. It works according to the XML rule specification and provides the following functionalities:

- Load the XML file of the AXCP rule from disk and generates an AXCP memory representation of rule (*AxCPRule* object)

AXRule Saver – It is the module for saving an XML representation of the rule on disk. It works according to the XML rule specification. It provides the following functionalities:

- Save the XML representation of the rule by replacing the existing one

- Save as function for saving the XML representation of the rule with a name



Both modules was implemented by using an abstract class called *AXRuleVisitor* as reported in the UML diagram. This solution allows building an *AXRuleLoader* and an *AXRuleSaver* class that manage different types of rules by implementing different *Visit* methods (see the class diagram reported below). Both classes are related to a DOM Document in order to perform the necessary read/write operations on an XML file. The XML representation of a rule is stored in the *DOMDocument* class from which it is possible to build the memory representation of the AXCP rule. The *AXRule* class provides a *Load* and *Save* method and a virtual method *Visit* that have to be redefined in the *AXCP Rule* class. In this way, the *Visit* method of AXCP Rule calls the *Visit* method of *AXRuleLoader* on the AXCP Rule object by using the *this* pointer.

Implementation was realised in C++ MSVC7 and supported by wxWidgets ver. 2.4.2 and XERCES 2.6.0 libraries.

4.5.1 AxRuleVisitor Class

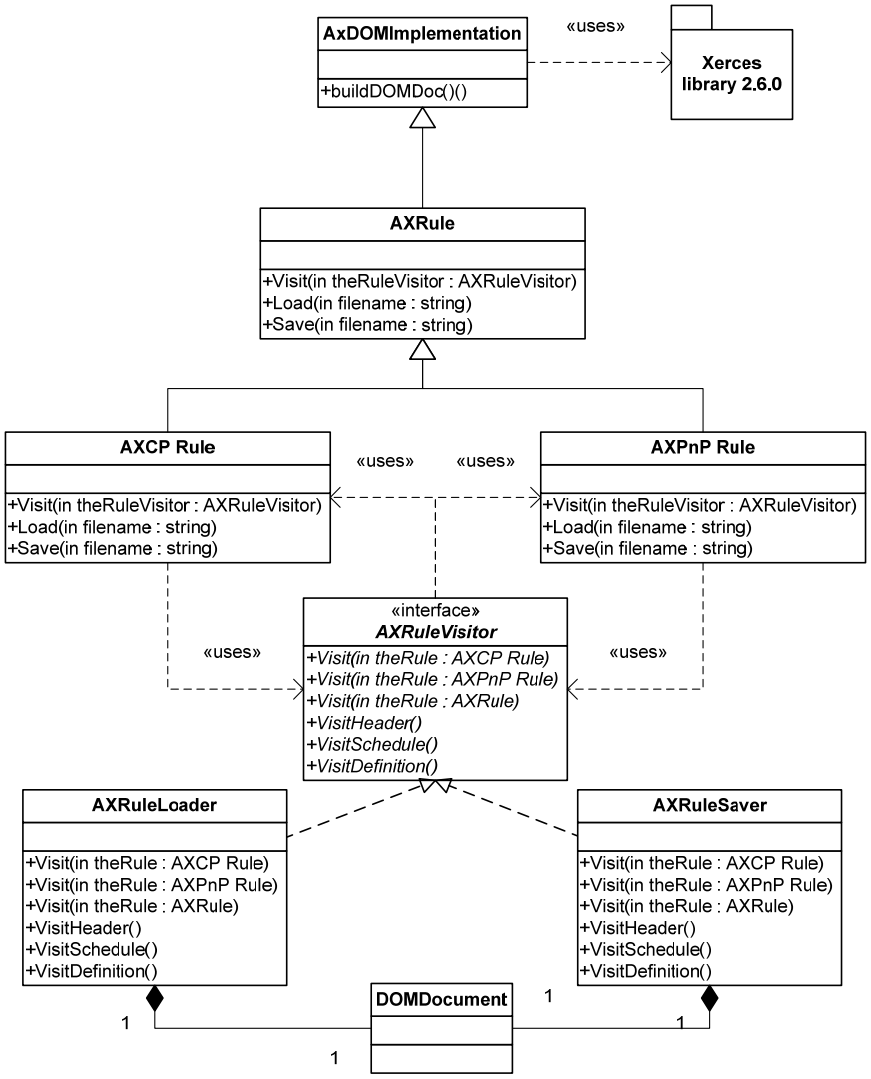
This class defines an abstract class for a Visitor. This class allows building a *Loader* and a *Saver* that manage different types of rules by implementing different *Visit* methods. The loader and saver visitor were implemented as specialized class of AxRuleVisitor.

4.5.2 AxRuleLoader Class

This class is a specialisation of the **AxRuleVisitor** class and implements a visitor for loading both **AxRule** and **AxCPRule** XML file. Since **AxRule** contains only *Header* and *Schedule* section, the visitor loader for such rule will read only *Header* and *Scheduler* sections. The visitor loader for the **AxCPRule** will call the visitor loader of the **AxRule** and then will read the *Definition* section. Methods of this class work polimorphically.

4.5.3 AxRuleSaver Class

This class is a specialisation of the **AxRuleVisitor** class and implements a visitor for saving both **AxRule** and **AxCPRule** XML file. Since **AxRule** contains only Header and Schedule section, the visitor saver for such rule will write only *Header* and *Scheduler* sections. The visitor saver for the **AxCPRule** will call the visitor saver of the **AxRule** and then will write the *Definition* section. Methods of this class work polimorphically.



UML Class Diagram of the AxRuleLoader and AxRuleSaver modules

5 AXCP Rule Editor (DSI)

Module/Tool Profile		
AXCP Rule Editor		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)	Approved	
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	Executable	
Single Thread or Multithread	Multi-Thread	
Language of Development	C++	
Platforms supported	Microsoft Windows	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/source/ruleeditor/ https://cvs.axmedis.org/repos/Framework/include/ruleeditor/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.axmedis.org/repos/Applications/ruleeditor/	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	Yes	
Usage of the AXMEDIS Error Manager (yes/no)	NO	
Major Problems not solved	-- --	
Major pending requirements	Intellisense, wordCompletion --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
AXMEDIS Workflow manager	Command and reporting	Web Service
AXCP Rule Engine		Web Service
Formats Used	Shared with	format name or reference to a section

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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
wxWidgets 2.4.2	wxWidgets 2.4.2 C++ library	LGPL
XERCES-C++	XERCES 2.6.0	LGPL
FL C++ Lib contribution to wxWidgets Lib	wxWidgets 2.4.2 C++ library	LGPL
STC C++ based on Scintilla editor, contribution to the wxWidgets Lib	wxWidgets 2.4.2 C++ library	LGPL
wxMozilla	wxWidgets 2.4.2 C++ library and Mozilla	LGPL

5.1 General Description of the Module

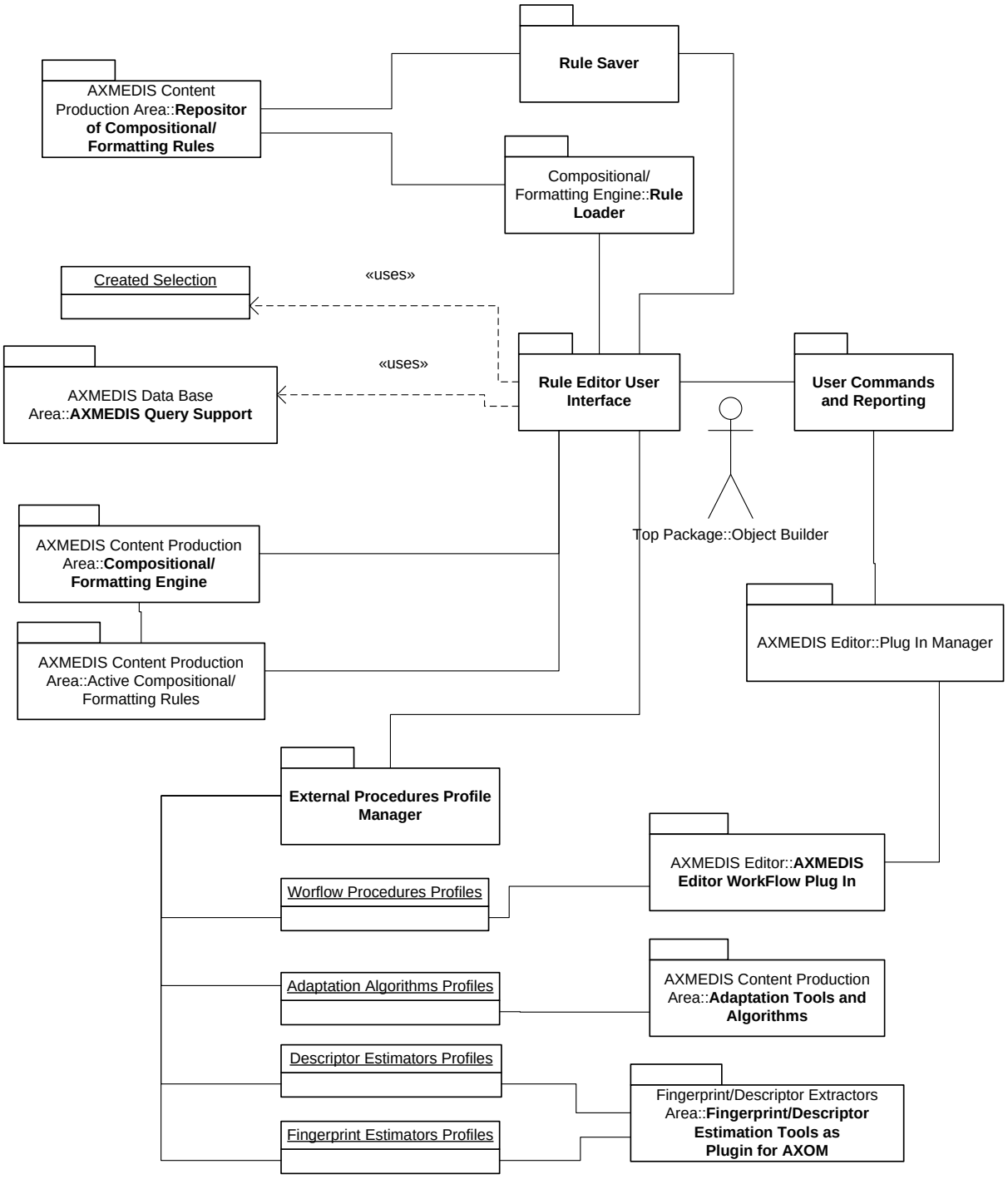
The AXCP Rule Editor is the tool that permits to obtain well formed rules by the means of both graphic and scripting tools. The AXCP Rule Editor is launched by the Workflow Web User Interface when the user/Rule producer clicks on a specified button in the Web page. During the launching an incomplete Rule XML is be sent to the AXCP Rule Editor. Some information could be already available in the header of rule (rule type, AXRID, comments etc…) and in the schedule. The Rule Editor is able to interact with an instance of the Rule Executor derived from the AXCP Rule Engine in order to perform debugging and feasibility check on rules. The main capabilities provided by the editor are:

- AXCP rule editing
- AXCP Rule Activation
- Internal textual editor for writing the Javascript code
- Debugging support to test and verify the Javascript code
- Communication support with the AXCP Rule Engine
- Communication support with the AXMEDIS Workflow Manager
- Selection editing and testing
- Metadata mapping

The specific components and aspects regarding protection and DRM can be found in section 5.2 of part H of the AXMEDIS framework specification “Protection Rule Editor”.

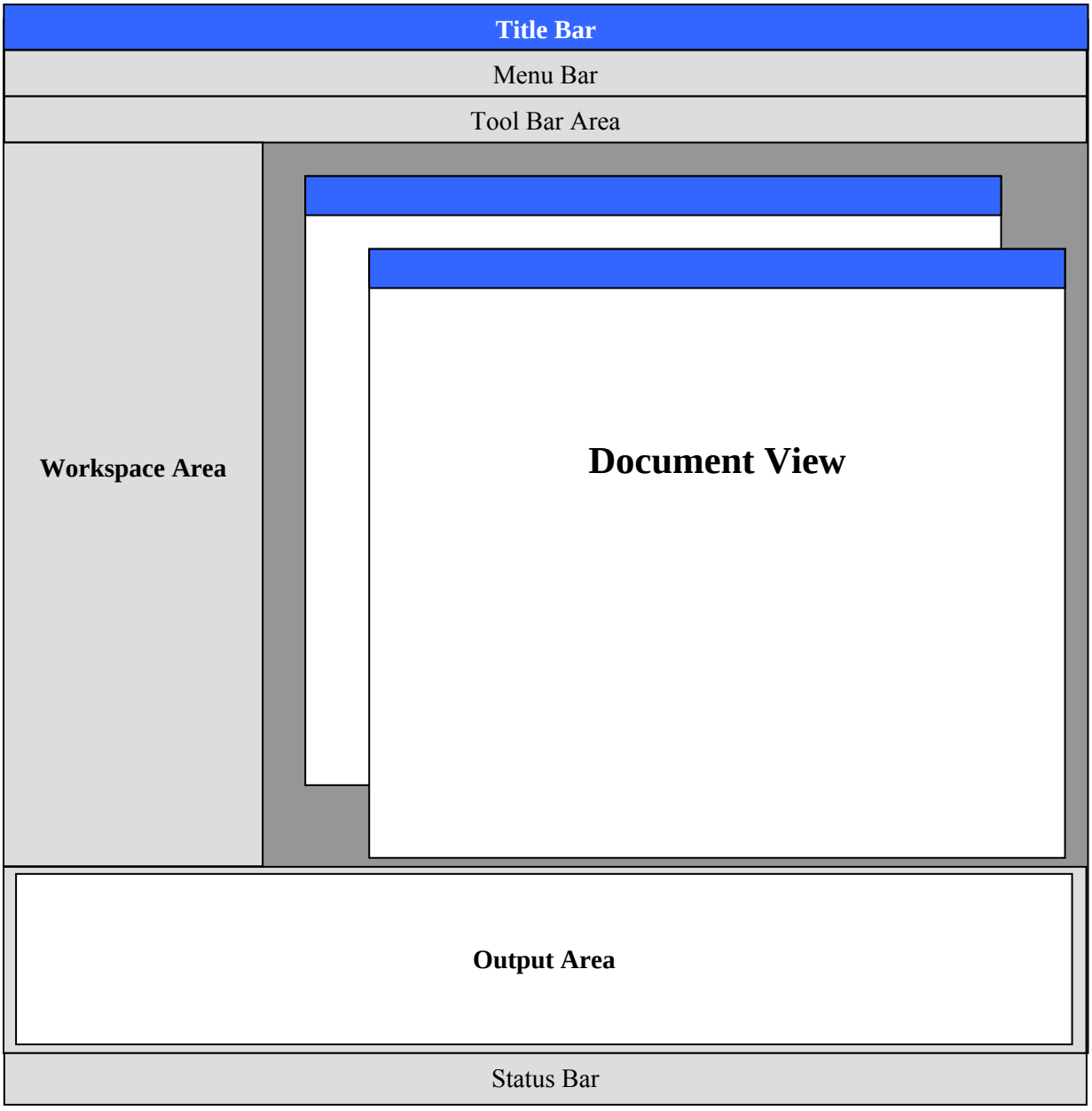
5.2 AXCP Rule Editor UML description

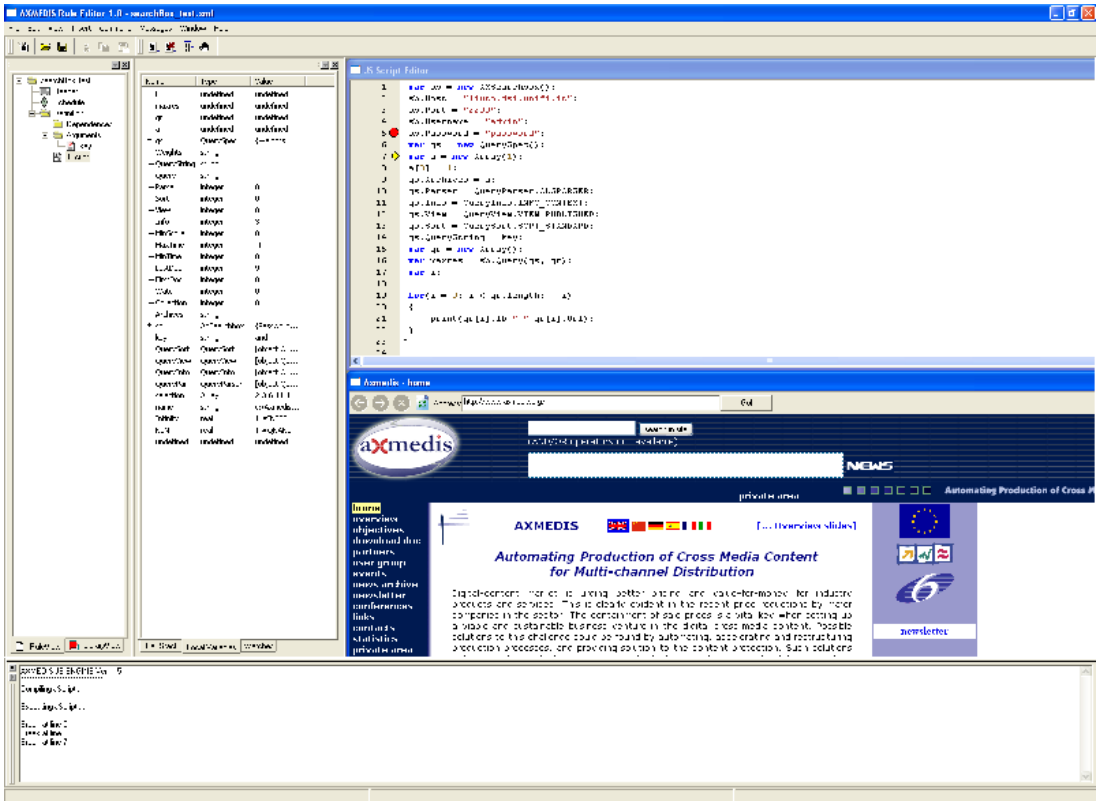
Compositional/Formatting Rule Editor
To be taken as an example of others Rule/Selection EDITOR



5.3 AXCP Rule Editor User Interface (DSI)

The AXCP Rule Editor is an IDE and multithread application and has been developed by using the wxWidgets ver. 2.4.2 library. This should allow having a multiplatform rule editor.
The AXCP Rule Editor GUI has been designed as MDI window that manages a rule document a time. It has been developed to provide a set of tools and views to help the user during the editing and production of rule. It hosts an instance of the AXCP Rule Executor in order to provide functionalities for debugging, testing and validating the script code associated with a rule. To help the user in writing rule, the editor has been equipped with an Help on line and an area where the user can access to a library of functions provided by AXMEDIS plugins and Javascript AXMEDIS classes and utilities. The main architecture of GUI is based on the following structure:





Main snapshot of theAxmedis Rule Editor GUI

5.3.1 The Menu Bar

The menu bar is constituted of the following entries:

File Edit View Insert Commands Debug Messages Window ?

File

- **New** – Create a new rule document
- **Open** – Open a AXCP rule in the Rule Editor
- **Close** – Close the current rule document
- **Save** – save the current rule using the current file name
- **Save as** – save the current rule by name
- **Import JScript** – import a script in the rule
- **Export JScript** – export the script on file system
- **Properties** – it shows a report on the
- **Page setup** – allow preparing the page for printing document
- **Print preview** – open the print preview dialog
- **Print** – send the document to the printer
- **Recent Files** – History of files
- **Exit** – Quit the editor

Edit

- Undo – the undo command
- Redo the redo command
- **Copy** – copy a text selection in the clipboard
- **Paste** – paste a text selection available in the clipboard
- **Cut** – delete and copy a text selection in the clipboard
- **Delete** – delete a selected item
- **Find…** – Search a word in the text

- **Find next** – search again for a new location of the current text
- **Replace…** – replace a word with another
- **Replace again** – replace again the word with a new entry
- **Match brace** – match the brace
- **Go to…** – go to a specific line text
- **Advanced menu**:
  - **Indent increase**
  - **Indent reduce**
  - **Overwrite mode**
  - **Wrap mode**
  - **Show line endings**
  - **Show indent guides**
  - **Show line numbers**
  - **Show long line markers**
  - **Show whitespace**
- **Select All** – select all content
- **Select line** – select the line where cursor is blinking

View

- **Workspace** – It opens the Workspace area
- **Output –** It opens the Output area
- **Debug Monitor –** It opens the debug window
- **Preferencese –** it opens the configuration dialog

Insert

- **Selection –** Adds a selection item in the rule
- **Parameter –** Adds a parameter item in the rule
- **Script -** Adds a script item in the rule
- **Dependency -** Adds a dependency item in the rule

AXCP Script

- **Var –** Adds a the statement for a jasvascript variable declaration
- **Function –** Adds a the statement for a jasvascript function declaration
- **Statement -** Adds a javascript statement
  - **If -** Adds an if statement
  - **If…else -** Adds an if-else statement
  - **switch -** Adds a switch statement
  - **do…while -** Adds do while statement
  - **while -** Adds a while statement
  - **for -** Adds a forf statement
  - **for…in -** Adds a for…in statement
  - **try…catch –** Add a try catch statement

Commands

- **Axcp Grid**
  - **Activate** – It is the activate rule command and will allow sending the current rule to the scheduler. A connection with the Rule Engine Scheduler will be open in order to perform the activation of rule in the Scheduler.
  - **Install** – It is the install rule command and will allow sending the current rule to the scheduler. A connection with the Rule Engine Scheduler will be open in order to perform the installation of rule in the Scheduler.

- **Get Rules** – It is the get rules command and will allow retrieving all rules installed in the scheduler. A connection with the Rule Engine Scheduler will be open in order to perform the request of rules to the Scheduler.
- **Run Rule** – It is the run rule command and will allow asking for running the current rule to the scheduler.

- **Find Rule…** – Allows making queries to the rules repository of the Rule Editor
- **Rules List…** – Shows the list of rules inside the repository of the Rule Editor
- **Check rule** – Tests the feasibility of the rule (like a compiler plus some tests on AXMEDIS objects and estimation of some parameters such as the files complexity and required workload)
-

Debug

- **Start** – Enter in the debug mode or if the script is stopped, continue execution until the script is finished, or a breakpoint is reached.
- **Stop** – Stop the script execution and close the debug mode
- **Step Over** - Executes the current line of the script, then pauses. This differs from the "Trace" command in that it will not step into functions and scripts that are called by the current line.
- **Trace Into** – Executes the current line of the script, then pauses. This differs from the "Step" command in that if the current line calls a function, or another script, the debugger will trace into the called function or script.
- **Insert/Remove Breakpoint** – Set a breakpoint on the currently selected line of the script code. Every time the selected line is reached, the debugger will stop. Clear a breakpoint from the currently selected line of the script code.
- **View Breakpoints list** – Open the debug window showing all breakpoints in the script code.

Tools

- **Selection Editor**– Open the Selection Editor window
- **Metadata Mapper** – Open the Metadata Mapper

Workflow

- **Pending rules List -** Displays the list of pending requests sent by the AXMEDIS Workflow Manager
- **Notify activity completion –** it open the dialog for notifying the completion of the activity to the AXMEDIS Workflow Manager

Window (provided automatically by the MDI GUI)

- **Cascade**
- **Tile Horizontal**
- **Tile Vertical**
- **Next –** Activate the next document view
- **Previous -** Activate the previous document view
- **Arrange Icons –** Arrange the all minimised document views
- **Close All –** Close all document views
- **Windows list**

Help

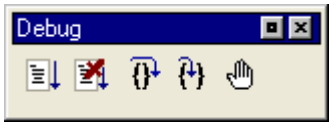
- **Help** – Open the internal Help
- **Registration –** Open the internal browser to the page for registering the tool
- **Import User Certificate** – Import the certicate after the registration
- **Tool certification** – Allow certifying the tool
- **About** – Information about the authors, version, etc

5.3.2 ToolBar Area

The toolbar area hosts a set of icon buttons that allows calling functions without accessing to the menu bar. The toolbar area is based on dockable toolbars and allows the dynamic customisation adding or removing sub-toolbars. For this end the editor provides sub-toolbars for:

- *Standard* functionalities such as:
  - *New rule document* – Create a new empty rule
  - *Open from disk* – Load a rule from disk
  - *Save* – Save a rule on disk
  - *Cut*
  - *Copy*
  - *Paste*

- *Debug* . It provides main controls for:
  - *Start Debug*
  - *Stop Debug*
  - *Step Over*
  - *Trace Into*
  - *Add/Remove Breakpoint*



- *AXCPGrid* . It provides main controls for:
  - *Install & Activate rule*
  - *Install rule without activation*
  - *Get Rules from GRID*

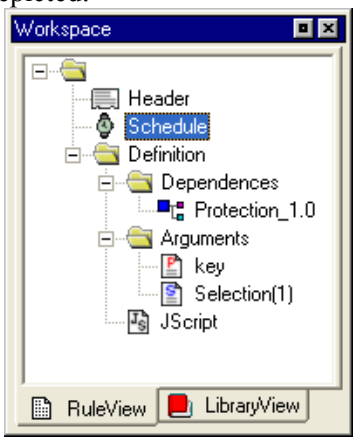
- *AXCP Script* . It provides main controls for:
  - *Reduce indent*
  - *Increase indent*
  - *Set selected rows as comment*
  - *Remove comment from selected rows*
  - *Var (JS statement)*
  - *Function (JS statement)*
  - *Code (statement list)*
  - *Search in scripts*
  - *Search history*

5.3.3 Workspace Area

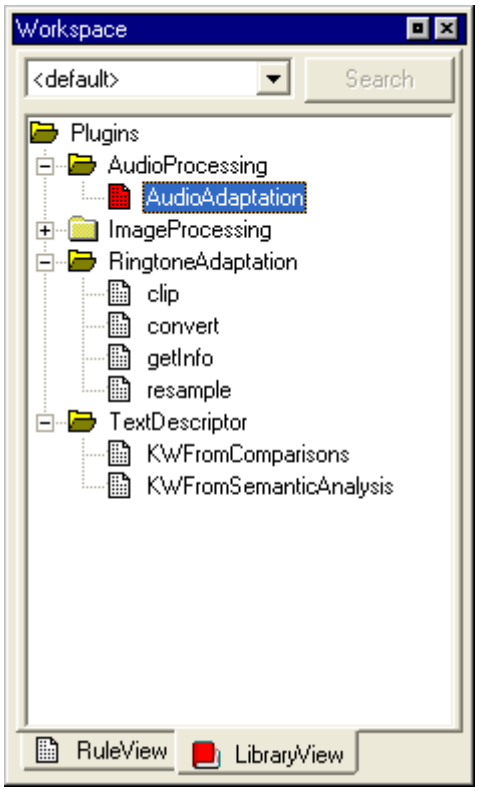
This is a resizable panel and includes a notebook control constituted of the following view items:

1. **Rule View –** In this area the structure of rule is displayed. It will be realised by using a Tree control that will permit to show and browse components according the rule XML schema.

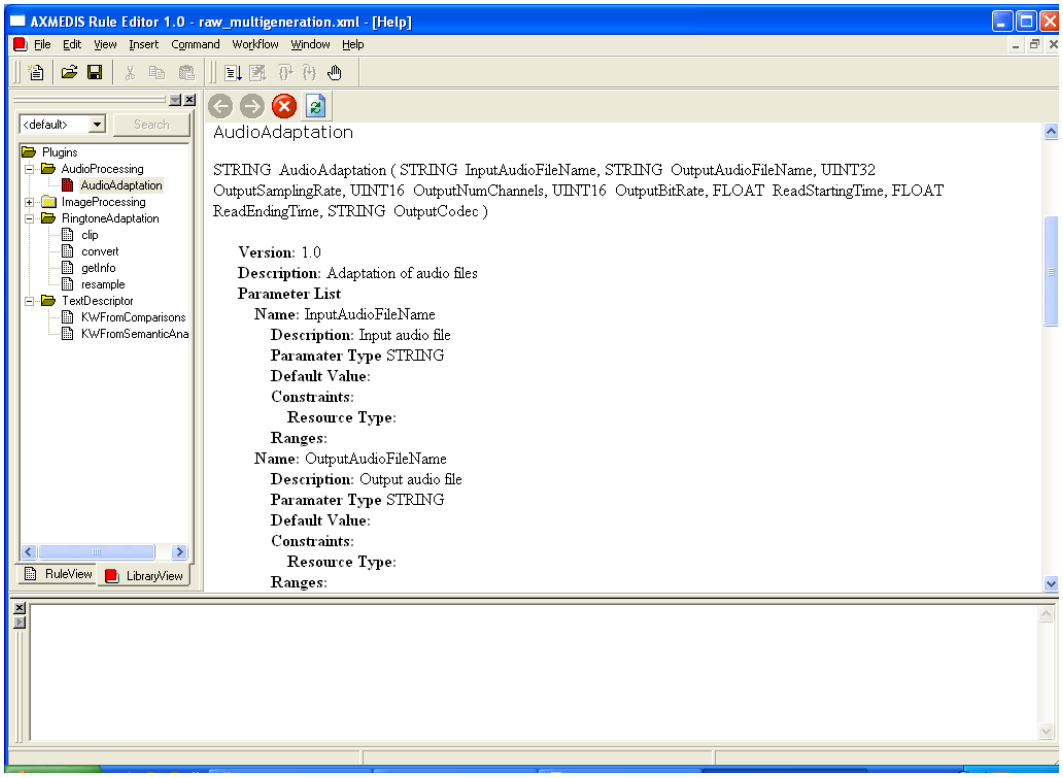
A dynamic popup menu will be available for a quick access to functions that will allow the quick management of items (edit and view metadata, delete,…). Appropriate icons will be also drawn in order to identify intuitively components of rule in the tree control view. In the following picture the structure of the Rule View area is depicted:



2. **Library view** – It is an on-line book that could be used as help by the user. It displays the set of functionalities provided by the Plugins installed and automatically detected by the editor. It is a tree control that permits to show and browse plugins module and the functionalities that they provide according to their profile.

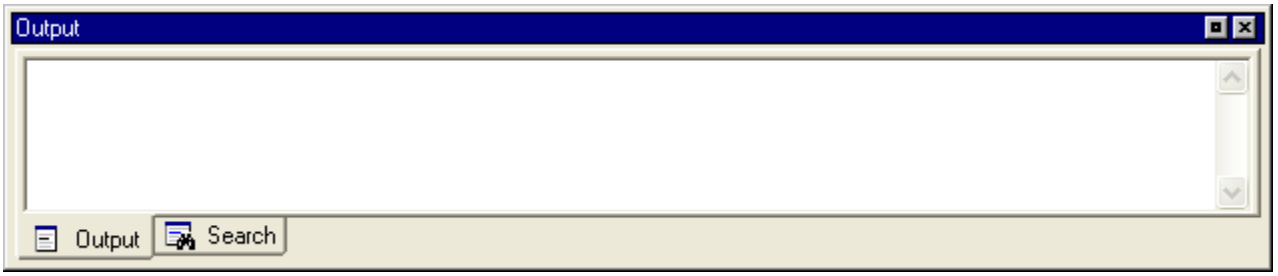


The user can see the documentation associated with each selected function by interacting with each item of the tree by double clicking or accessing to a contextual popup menu. The Search function is available to save time in finding function, it works by using mime type of signature parameters as serching criteria The selected documentation is displayed in the **Text/Html document view.**



5.3.4 Output & Search Area

This is a text control where messages, textual description, errors, debugging info, alert, etc… are displayed.



5.3.5 Tools, Viewers and Editors

Some different types of tools and editors has been designed for visualizing and/or editing different types of document.

1. **Javascript editing window** – This is the window client where the user will be able to write the script code. It will be based on a multiline text control where it will be possible to edit the script. The textual editor will support some facilities such as:
    - Auto completion of words - a window listing possible completions for strings the user has typed
    - Syntax highlighting – keywords will be colourised
    - Brace highlighting
    - Folding/Hiding - making lines invisible or visible. It shows or hides a range of lines.
    - Multiple views - to have multiple views of the same Document. (Split view)
    - Breakpoint insertion/removal – to control the code in the debugging session
    - Visualisation of line numbers

```
184  //=============================================================================
185  // implementation
186  //=============================================================================
187
188  IMPLEMENT_APP (App)
189
190  //-----------------------------------------------------------------------------
191  // App
192  //-----------------------------------------------------------------------------
193
194  bool App::OnInit () {
195
196      wxInitAllImageHandlers();
197
198      // set application and vendor name
199      SetAppName (APP_NAME);
200      SetVendorName (APP_VENDOR);
201      g_appname = new wxString ();
202      g_appname->Append (APP_VENDOR);
203      g_appname->Append (_T("-"));
204      g_appname->Append (APP_NAME);
205
206      // initialize print data and setup
207      g_printData = new wxPrintData;
208      g_pageSetupData = new wxPageSetupDialogData;
209
210      // create application frame
211      m_frame = new AppFrame (*g_appname);
212
213      // open application frame
214      m_frame->Layout ();
215      m_frame->Show (true);
216      SetTopWindow (m_frame);
217
218      return true;
219  }
220
221  int App::OnExit () {
222
223      // delete global appname
224      delete g_appname;
225
226      // delete global print data and setup
227      if (g_printData) delete g_printData;
228      if (g_pageSetupData) delete g_pageSetupData;
229
```

2. **Text/Html document view –** This is the window for the visualisation of the documentation provided by the help on line. It is opened when the user double clicks on a voice of the index in the *Library view* or when the internal help is called. It provides functionalities for browsing TXT or HTML pages. For example, all the information related to the description of a function selected from the *Library view* are shown in such window.

3. **Selection Editor -** It is an iternal editor that provides functionality for:
    a. Edit a selection
    b. Save/Load a selection
    c. Actualise the selection

For more details about the Selection editing see the DE3.1.2E Framework and Tools Specifications (Database and Gathering)

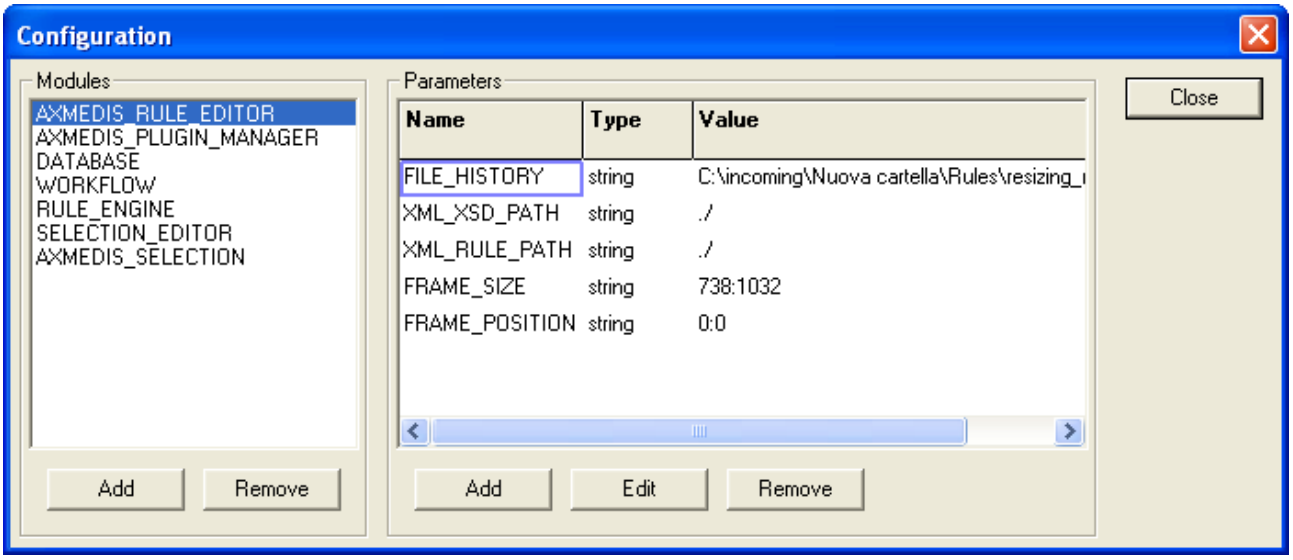
4. **Mapper Editor**

Mapper editor GUI will be an editor for Metadata Mapping AXEPTool. It will be the interface used to creates the map for incoming metadata translation. The GUI allows the user to decide which origin fields have to be converted in destination fields. The GUI will be invoked by the AXCP Editor.
For more details see document DE3.1.2F Framework and Tools Specifications (AXEPTool and Progr. and Pub.).

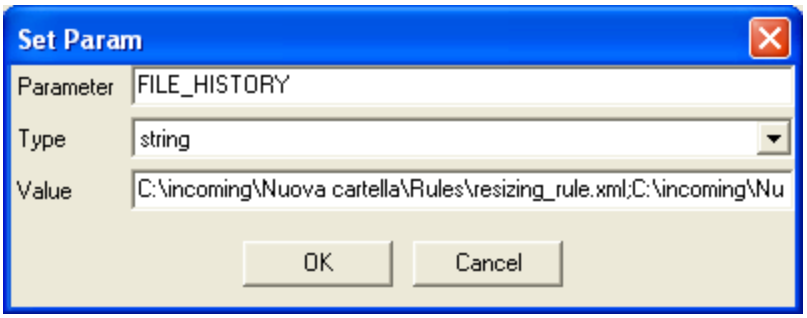
5.3.6 Interactive Dialogs

The editor provides a set of dialogs for facilitate the editing of a rule. To this end, the following dialogs were designed:

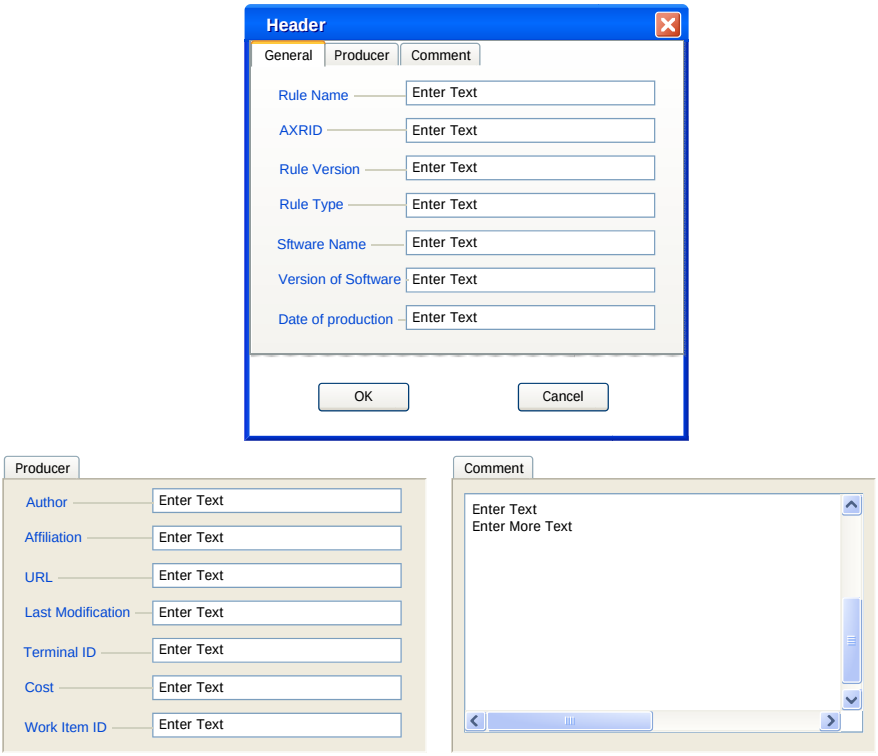
1. **Configuration Dialog:** The AXCP Rule Editor allows accessing to the configuration dialog when it is necessary to modify the configuration file. The configuration Dialog is the same dialog used in the AXMEDIS Editor, it provides the list of modules and for each of them the set of parameters. The user can modify them by means of the "Add", "Edit", "Remove" functions.



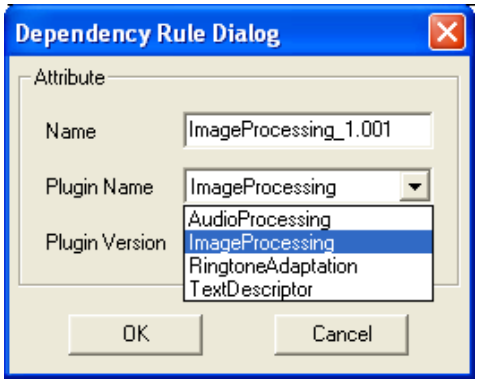
The "Edit" function opens the "Set Param" dialog where it is possible to change values.



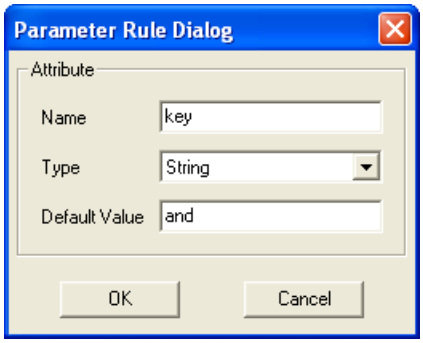
2. **Header Rule Edit Dialog –** This is a tabbed dialog that allows filling fields of the header section. The dialog is designed as an OK/Cancel modal dialog in a notebook style with General, Producer and Comment tab where the list of items to edit are displayed.



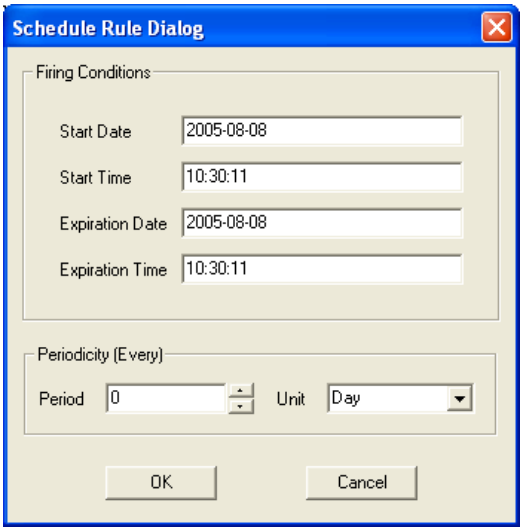
3. **Dependency Edit Dialog -** This dialog allows filling fields for a dependency item. The dialog is designed as an OK/Cancel modal dialog and displays the list of item to edit. The dialog shows the list of plugin installed in order to facilitate the choice.



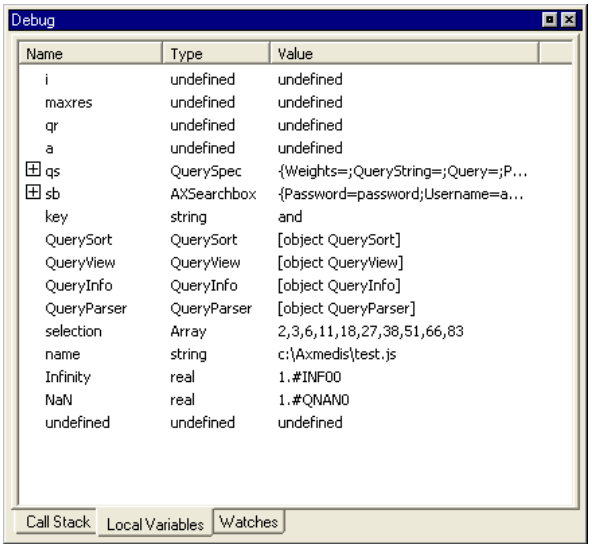
4. **Parameter Edit Dialog -** This dialog allows editing/filling fields for a parameter item. The dialog is designed as an OK/Cancel modal dialog and displays the list of items to edit.



5. **Schedule Edit Dialog –** This dialog allows filling/editing fields for a schedule item. The dialog is designed as an OK/Cancel modal dialog and displays the list of items to edit.



6. **Local Variables –** This is a page of the Debug dockable window that displays variables and instances of objects allocated by the script. They are displayed as tree list control with folding-unfolding capability for displaying the list of attributes of the object instance.



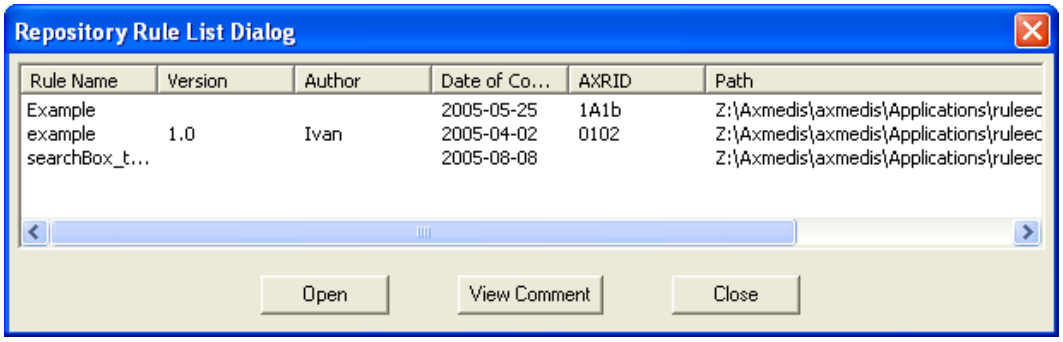
7. **Call Stack –** This a page of the Debug dockable window that displays stack of functions calls. It display the script name and the line of the call. Double clicking on an entry level of the stack allows focusing the view of the script where the corresponding call is located.

8. **Breakpoints -** This a page of the Debug dockable window that displays the list of breakpoints inserted in the scripts: each breakpoint is associated with the name of the script and relative line number and status (Enabled or Disabled). Double clicking on a breakpoint allows focusing the view of the script where the breakpoint is placed.

9. **Repository Rule List Dialog -** The *Rule List* command opens a rules list modal dialog displaying all rules stored in the repository of the AXCP Rule editor. In this window, the list of rulesis organised in a table built on the following subset of metadata:
    - *Rule Name*
    - *Rule Version*
    - *Author*

- *Date of composition*
- *Rule ID (AXRID)*
- *Path*

The user can select a specific rule in order to open it in the rule editor. Such operation is possible by pushing the *Open* button or double clicking on the line of the chosen rule. The user visualizes the comment associated with rule by pushing the *View Comment* button, the comment is displayed in the *Output Area* of the Rule Editor. Otherwise the user cancels the operation by closing the dialog or pushing the *Close* button.



10. **Find –** This will be a common/standard find text no modal dialog where the user will put strings he wants to search in the text.

11. **Go To –** This will be an OK/Cancel no modal dialog where the user will put the number of the line where he wants to set the cursor

12. **Find Rule dialog –** The find rule command will open a dialog that will allow setting a query in order to search a specific rule inside the repository. The query will be built on the following set of metadata:
    - *Rule Name*
    - *Rule Version*
    - *Rule Type*
    - *Date (day, month and year)*

Two logical operators will be available to make query: OR and AND.

In event of some metadata missing, the query on the repository will be done with the available metadata. An empty query will be not executed, this will be controlled by activating the *OK* button when at least a metadata has been input.
Clicking on the *OK* button will start the search in the rule repository. The *Cancel* button aborts the operation.

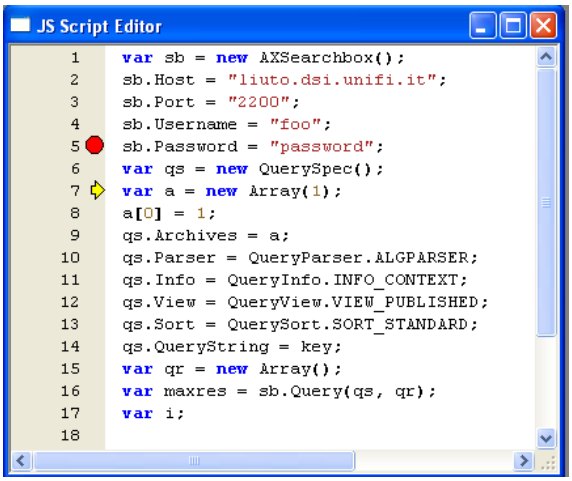
5.3.7 Debugging Rules (DSI)

The editor provides the debug mode for rule debugging. The debugging mode is possible using an instance of the Script Executor inside the AXCP Rule Editor..To this end the Script Executor is used in the Debugging mode and provides functionalites and data structure to:

- maintain the list of breakpoints to put in the code
- put traps in the code corresponding to breakpoints
- update call stack window
- call the rule engine and initialise it to the debugging mode
- manully control the execution of the script
- print ouput and internal errors when script runs.

The debug mode provides the following set of manual functions:

- **Start Debug/Next Breakpoint** – Enter in the debug mode or if the script is stopped, continue execution until the script is finished, or a breakpoint is reached.
- **Stop Debug** – Stop the script execution and close the debug mode
- **Step Over**- Executes the current line of the script, then pauses. This differs from the "Trace" command in that it will not step into functions and scripts that are called by the current line.
- **Trace Into**– Executes the current line of the script, then pauses. This differs from the "Step" command in that if the current line calls a function, or another script, the debugger will trace into the called function or script.
- **Skip** - Skips over current line without executing it. The script resumes the execution on the subsequent line.
- **Set Breakpoint** – Set a breakpoint on the currently selected line of the script code. Every time the selected line is reached, the debugger will stop.
- **Remove Breakpoint** - Clear a breakpoint from the currently selected line of the script code.
- **Remove All Breakpoint** - Clear a all breakpoint in the script code.



The debug mode provides a set of icons to facilitate the useage and the interaction with the Script Editor:

- A red filled circle indicates an Enabled breakpoint
- A red empty circle indicates a Disabled breakpoint
- A yellow arrow indicates the line that will be executed.

5.3.8 User Commands and Reporting - AXMEDIS Workflow Manager interaction (DSI)

It provides communication support from/to the AXMEDIS Workflow Manager. Such services are divided in commands for the user and reporting.

- Messages coming from the AXMEDIS Workflow Manager:
  - Description of the work to perform
  - AXRID to associate with the rule to be prepared
  - Rule type
  - Schedule information

    - Some other information related to header of the rule (according to the XML schema)
- Reporting to the AXMEDIS Workflow Manager
    - Notification (end work with success)
    - Exception

Communication with the AXCP Rule Engine

- Commands to the AXMEDIS Rule Engine:
    - Activate rule
    - Install rule in the scheduler (xml file transfer)
- Notification, messages, files and exception returned by the AXMEDIS Rule Engine.

The communication between the AXCP Rule Editor and the Engine is based on the same protocol used by the Workflow Manager for communicating with the AXCP Rule Engine. It is Web Service-based.

5.3.9 External Procedures Profile Manager (DSI)

To enable AXMEDIS to be a flexible structure, which can be extended according to the specific user needs, plug-ins can be easily be integrated into the AXMEDIS framework. The plug-ins are handled by the Plug-In Managers (AXMEDIS Editor::Plug-In Manager and the Collector Engine::Collector Plug-In Manager).

The Plug-in Managers takes care about the installation, registration, and loading of plug-ins. Different kinds of plug-ins are supported, including:

- Data-manipulation plug-ins shall be able to modify AXMEDIS object structure, i.e. plug-ins which shall be able to delete or move existing components, insert new components, etc…
- Metadata show/manage plug-ins shall be used by Metadata View to adequately display and modify user-defined sets of metadata;
- Metadata production shall be able, through AXMEDIS object (and parts thereof) analysis, to produce metadata to be included into the object;
- Configuration plug-ins shall be used by AXMEDIS Editor Configuration Manager to manage and display specific configuration information;
- Workflow plug-ins which shall permit interaction of AXMEDIS Editor with AXMEDIS Workflow subsystem;
- Protection plug-ins, which contain protection algorithm enriching the set of those available for the Protection Processor;

To enable an effective management, a profile manager is responsible for the handling of the profile descriptions. This profile manager is very close to the Plug-In manager. It is responsible for loading the profiles from installed libraries. The relevant information includes:

- the category of the plug-in, e.g. content processing;
- the unique identifier of the plug-ins
- the signature of the plug-in
- data specific for the kind of plug-in
- the signature

As the external procedures profile is closely related to Plug-in Manager and the content processing algorithms details can be found in DE3-1-2-2-3 and in DE3-1-2-2-7, respectively.

5.4 Draft User Manual

5.4.1 Editing an AXCP Rule on AXCP Rule Editor

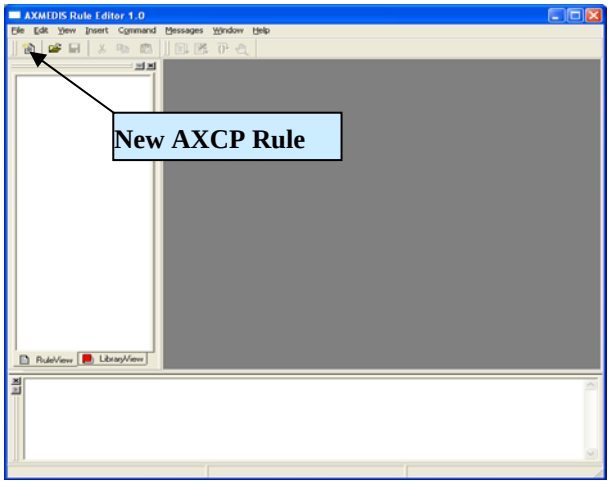
A content producer or manager wishes to create a content processing rule called AXCP Rule for manipulating/creating AXMEDIS multimedia objects.

MS Windows Firewall Alert at startup

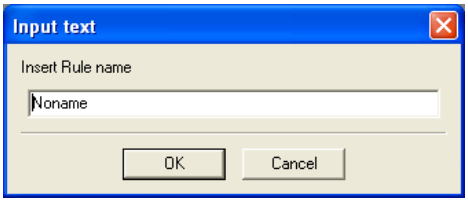
The picture bellow shows the Windows Security Alert Dialog. To run AXCP Tools, please unlock the application clicking on the "Unlock" button. This operation allows AXCP tools to use network services and run properly.

Creating a new rule

To create a rule, the user starts the AXCP Rule Editor and from the opening screen creates a new Rule by selecting "*New*" from the tool bar or using the File menu (File → New) or using the keyboard shortcut "Ctrl+N" (see Screenshot 1). On requesting a new rule, the dialog box pops up and the user enters the name of the draft AXCP Rule and selects "OK" (see Screenshot 2).

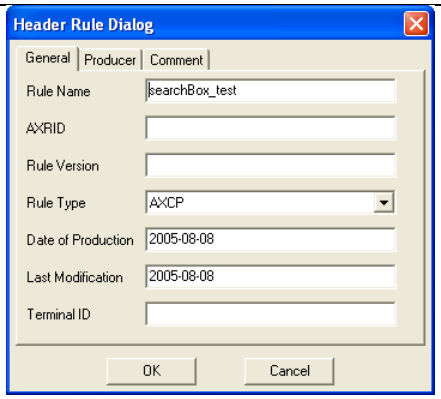
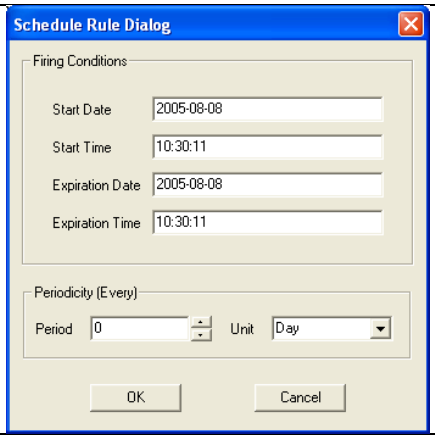
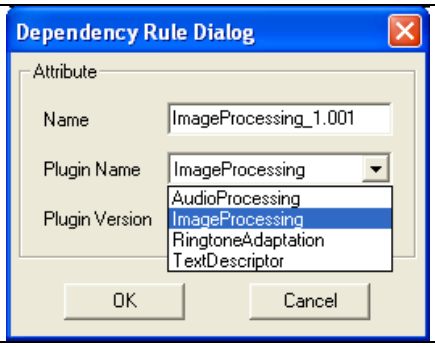


OPEN A NEW RULE



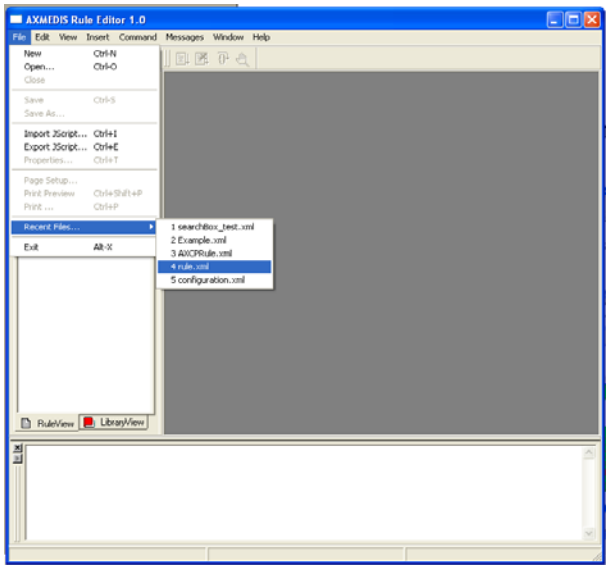
INPUT RULE NAME DIALOG

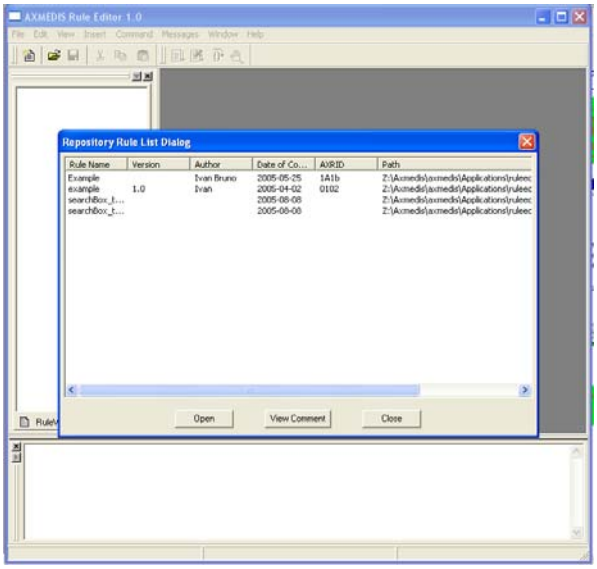
After the rule name has been entered, the new rule is ready for editing with the '*tree view*' used as a workspace and the editing dialogs for editing the rule details and making a rule schedule.

	**Header dialog** This is the dialog that allows filling fields of the header section. The dialog is an OK/Cancel modal dialog in a notebook style with *General*, *Producer* and *Comment* tab where the list of items to edit is displayed. **Note**: The AXRID field is read only and it is pre-filled with the identification code assigned by the rule editor
	**Schedule dialog** This dialog allows filling/editing fields for a schedule item. The dialog is as an OK/Cancel modal dialog and displays the list of items to edit. The dialog allows setting the start date and time, the expiration date and time, and the conditions of periocity (how may times in term of unit such as days, week, month and so on)
	**Dependecy dialog** This dialog allows filling fields for a dependency item. The dialog is an OK/Cancel modal dialog and displays the list of available plugins in order to facilitate the choice.

Loading an existing rule

Instead of creating a new rule, the user may wish to edit an existing draft rule. The user may have saved it as a file somewhere or saved it in the AXCP Rule Repository. By selecting '*open*' or '*Rule List*' or selecting a file in the history list ('*Recent Files…*'), as seen in screenshot 4 and 5, the rule editor can load an existing draft rule from the Repository or elsewhere on the system.





Open commands shows the commond Open dialog

The *Rule List* command opens a rules list modal dialog displaying all rules stored in the repository of the AXCP Rule editor. In this window, the list of rules will be organised in a table built on the following subset of metadata:

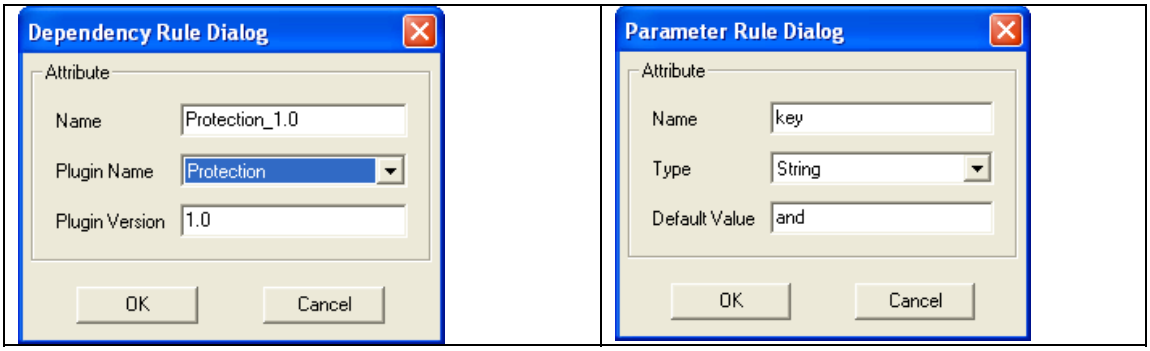
- *Rule Name*
- *Rule Version*
- *Author*
- *Date of composition*
- *Rule ID (AXRID)*

The user can select a specific rule in order to open it in the rule editor. Such operation is possible by pushing the *Open* button or double clicking on the line of the chosen rule. The user can visualize the comment associated with rule by pushing the *View Comment* button, the comment is displayed the *Output Area*. Otherwise the user can cancel the operation by closing the dialog or pushing the *Close* button.

Editing a rule

The user can use the set of dialog and text editor to edit the AXCP rule (as shown in screenshots), and edit rule data such as:

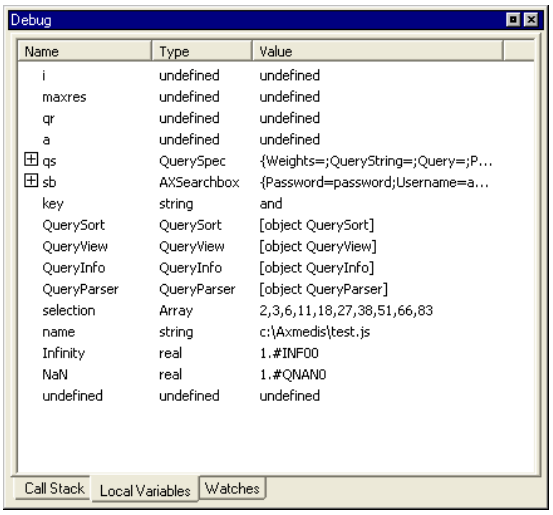
- Parameter dialog for editing the attributes of a rule parameter
- Dependency dialog for editing the attributes of a AXMEDIS PlugIn
- XML Selection Editor (XML viewer/editor for the XML representation of selections) and JavaScript Editor based on Scintilla Editor for text/javascript code editing. It provides full editing capabilities (copy, cut, paste, redo, undo, syntax highlighting, etc…), print preview, page setup and print functionalities, syntax highlighting, brace highlighting, folding/hiding of lines, breakpoint insertion/removal, visualisation of line numbers

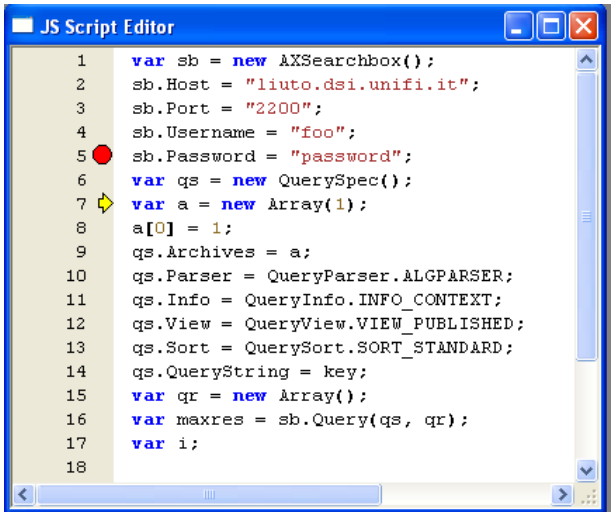


Dependency dialog	Parameter dialog

Debugging Rule functionalities

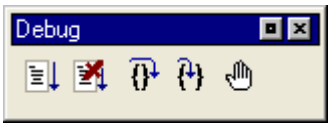
The editor provides functions to add/remove breakpoints (F9), start debug (F5), next breakpoint (F5), step over (F10) and step into (F11), stack calls monitoring, local variables visualization. In the output window a textual output provides internal errors or communication when script runs and during the debug session





Main navigation interface of a Debug session and markers:

- A red filled circle indicates an Enabled breakpoint
- A red empty circle indicates a Disabled breakpoint
- A yellow arrow indicates the line that will be executed.



Activating a rule

The current version of the prototype allows installing a rule in the rule engine and notifying the completeness of the rule to the workflow manager. The activation can be done by using the *Activate* function in the Command menu or manually as described in the scenario 2.

5.5 Configuration Parameters

In this section the set of parameters regarding the configuration of the editor are listed. Such parameters are grouped into modules as reported below:

5.5.1 AXMEDIS Rule Editor

Config parameter	Possible values
XML_RULE_PATH	it is the directory where the rule will be saved
XML_XSD_PATH	It is the directory where xml schema (XSD files) are stored
FRAME_SIZE	it is the information about the last width and height of the main frame
FRAME_POSITION	It is the information about the last position (x,y) of the main frame
FILE_HISTORY	It is the list of recent rule documents
REGISTRATION_PORTAL	It is the link to the Web Page of the registration Portal for user registration

5.5.2 Workflow Manager

Config parameter	Possible values
workflowUrl	It is the URL for the workflow plugin
gatewayUrl	It is the gataway URL for the workflow
WorkflowPluginId	It is the identificator of the workflow plugin (dll)

5.5.3 AXMEDIS Plugin Manager

Config parameter	Possible values
PLUGINS_PATH	It is the directory where the DLL of plug-ins with their profiles (workflow, adaptation, descriptor and fingerprint estimators) are stored.

5.5.4 AXMEDIS Database

Config parameter	Possible values
user	The user name for logging into Database
passwd	The passoword for logging into Database
LoaderWSEndPoint	It is the URL for the loader
HTTPPath	It the HTTP path
UploadPath	It is the Upload path
SaverWSEndPoint	It is the URL for the saver

5.5.5 AXMEDIS Rule Engine

Config parameter	Possible values
gatewayUrl	it is the URL of the AXCP Rule Scheduler/GRID

5.5.6 AXMEDIS Selection

Config parameter	Possible values
MAIN_QUERY_SUPPORT_WSDL	It is the URL of the WSDL for using the Main Query Support
SELECTION_ARCHIVE_WSDL	It is the URL of the WSDL for using the Selection Archive

Example of Configuration file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 3 U (http://www.xmlspy.com)-->
<Configuration xmlns="http://www.axmedis.org/configuration" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.axmedis.org/configuration configuration.xsd">
    <Module category="" id="AXMEDIS_RULE_EDITOR">
        <Parameter name="FILE_HISTORY" type="string">C:\incoming\Nuova
cartella\Rules\resizing_rule.xml;C:\incoming\Nuova
cartella\Rules\multigeneration.xml;C:\Axmedis\axmedis\Applications\ruleeditor\bin\win32\rt_test_rule.xml;\\MASTER\ex-
wedel\ilabs-lobster.xml;C:\Axmedis\axmedis\Applications\ruleeditor\bin\win32\ilabs-
lobster.xml;C:\PictureAndDoc.xml;C:\Noname.xml;C:\queryTest.xml;C:\axexamples\ruleditor\Rules\raw_multigeneration.xml;</P
arameter>
        <Parameter name="XML_XSD_PATH" type="string">./</Parameter>
        <Parameter name="XML_RULE_PATH" type="string">./</Parameter>
        <Parameter name="FRAME_SIZE" type="string">738:1024</Parameter>
        <Parameter name="FRAME_POSITION" type="string">0:0</Parameter>
    </Module>
    <Module category="" id="AXMEDIS_PLUGIN_MANAGER">
        <Parameter name="PLUGINS_PATH" type="string">./AxPlugin</Parameter>
    </Module>
    <Module category="" id="DATABASE" visible="true">
        <Parameter name="user" type="string">test</Parameter>
        <Parameter name="passwd" type="string">test</Parameter>
        <Parameter name="LoaderWSEndPoint" type="string">http://192.168.1.10:8080/LoaderSaverWS/Loader</Parameter>
```

```
        <Parameter name="HTTPPath" type="string">http://192.168.1.12/axdb-share</Parameter>
        <Parameter name="UploadPath" type="string">C:\wedelmusic\localdistributor\htdocs\axdb-share</Parameter>
        <Parameter name="SaverWSEndPoint" type="string">http://192.168.1.10:8080/LoaderSaverWS/Saver</Parameter>
    </Module>
    <Module category="" id="WORKFLOW">
        <Parameter name="workflowUrl" type="string">http://192.168.1.118:8080/Prove_WF</Parameter>
        <Parameter name="gatewayUrl"
type="string">http://192.168.1.118:8000/responseGateway/reChannel/reChannel.asmx</Parameter>
    </Module>
    <Module category="" id="RULE_ENGINE">
        <Parameter name="gatewayUrl" type="string">http://192.168.0.91:9000/</Parameter>
    </Module>
    <Module category="" id="AXMEDIS_SELECTION">
        <Parameter name="MAIN_QUERY_SUPPORT_WSDL" type="string">http://bellini-
mobile:8080/MainQuerySupportWS/mqs?WSDL</Parameter>
        <Parameter name="SELECTION_ARCHIVE_WSDL" type="string">http://bellini-
mobile:8080/UserSelectionArchiveWS/sa?WSDL</Parameter>
    </Module>
</Configuration>
```

5.6 Errors reported and that may occur

Error code	Description and rationales

6 AXCP Rule Engine (DSI)

Module/Tool Profile		
AXCP Rule Engine		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)	Proposed	
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	Executable	
Single Thread or Multithread	Multi-Thread	
Language of Development	C++	
Platforms supported		
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/source/rulescheduler/ https://cvs.axmedis.org/repos/Framework/source/ruleexecutor/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.axmedis.org/repos/Applications/rulescheduler/ https://cvs.axmedis.org/repos/Applications/ruleexecutor/	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	YES	
Usage of the AXMEDIS Error Manager (yes/no)	NO	
Major Problems not solved	-- --	
Major pending requirements	CPU usage balance, Activity planning based on Dead Line Monotonic	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
AXMEDIS Workflow manager	Command and reporting	Web Service
Formats Used	Shared with	format name or reference to a section
AXCP Rule Model		AXCP Rule Model

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Views Manager	C++	wxWidgets 2.4.2 C++ library
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
wxWidgets 2.4.2	wxWidgets 2.4.2 C++ library	LGPL
SpiderMonkey JavaScript Engine	SpiderMonkey JavaScript Engine ver. 1.6 by Mozilla	LGPL
XERCES-C++	XERCES 2.6.0	LGPL

6.1 General Description of the Module

In this section, the specification of the AXCP Rule Engine will be discussed. As already discussed, the content processing activity (production, protection and publication on AXEPTool) will be based on a unified and shared solution. In these terms, the AXCP Rule Engine will play the role of:

- AXMEDIS Compositional/Formatting Engine
- AXEPTool Loading Tool Engine
- AXEPTool Publication Tool Engine
- AXEPTool P2P Active Selection Engine
- The Protection Tool Engine

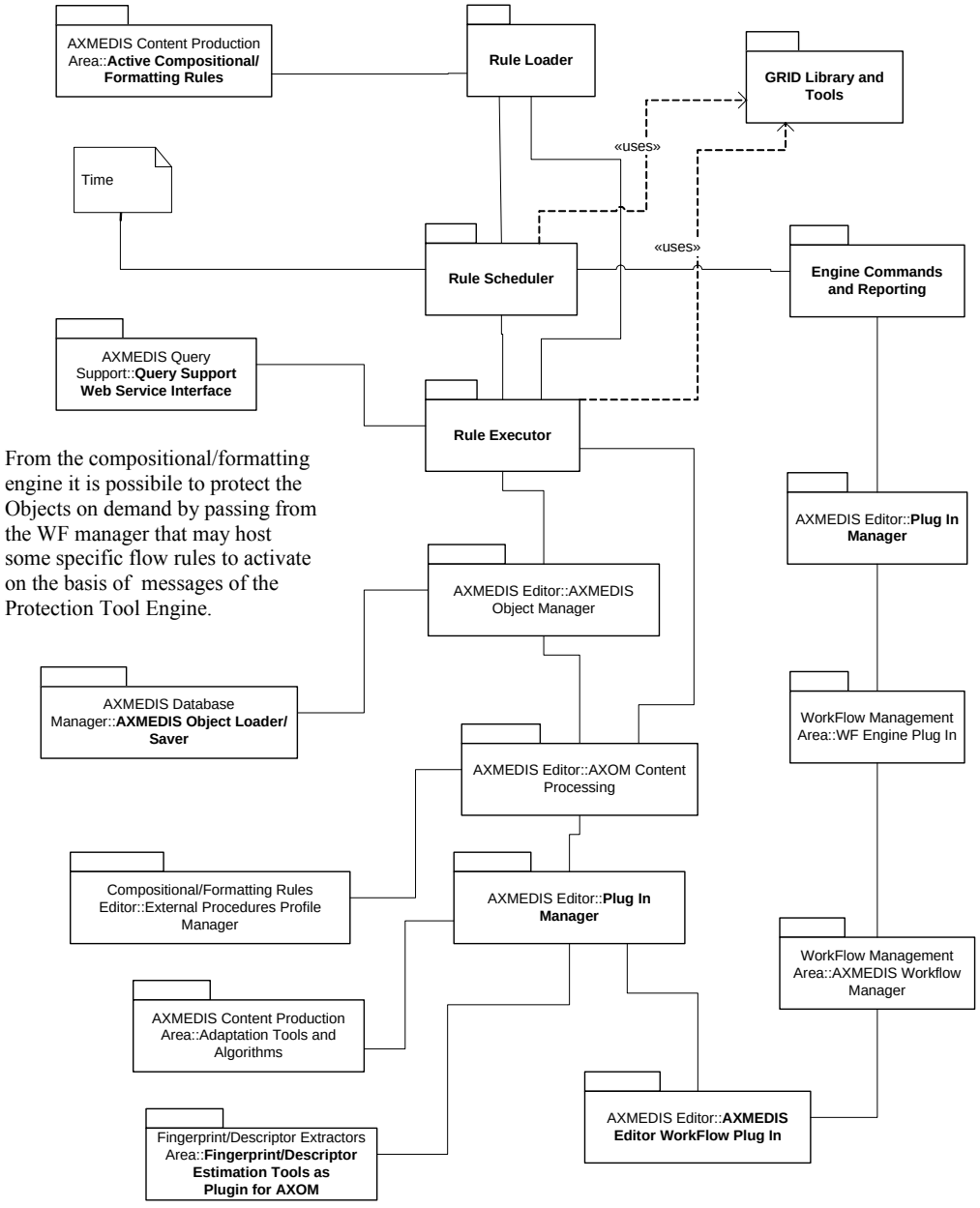
By delegating the processing activity to a single rule engine seems to be not the best solution since the amount of work and the dimension of data that the engine will have to manage is high. The main idea is to design a distributed environment of engines for the AXMEDIS object processing based on GRID. This solution will maintain advantages of a unified solution and allow enhancing the capabilities of the AXMEDIS content processing area by running rules in parallel.
According to the UML diagram, the AXCP Rule engine will be divided in two main components:

- **Rule Scheduler (Server Side)** – It consists of the Scheduler and Dispatcher. It performs the operations of rule firing, rule executor discovering and management, rules dispatching, communication with the AXMEDIS environment, etc….

- **Rule Remote Executor** (**Client Side**) – It is the executor of rules and consists of a script engine based on JavaScript (JS) SpiderMonkey released by Mozilla. It receives the JavaScript code associated with rule and performs the necessary operations for: Script preparation, JS Engine initialisation, JS Engine running script

The **Grid infrastructure** is realised by means P2P technology. For these reason both the Rule Scheduler and the Rule Executors host a P2P communication support.

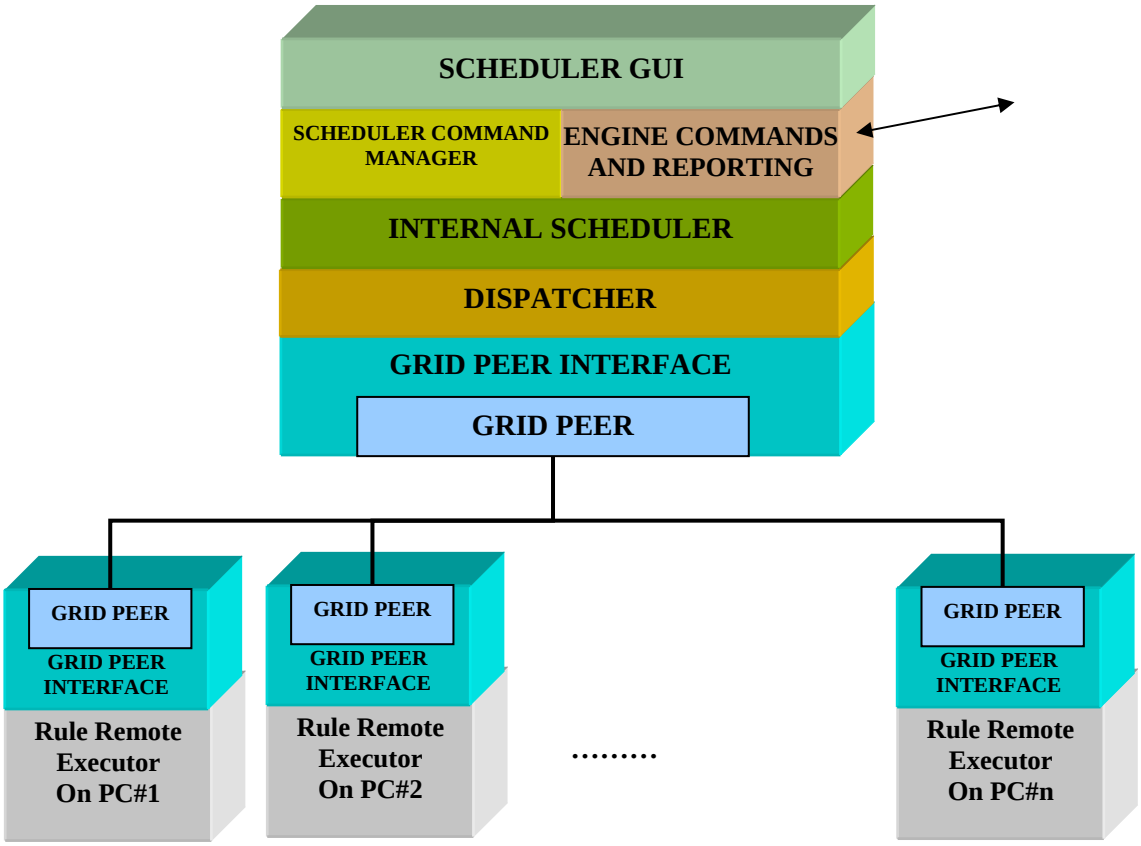
Compositional/Formatting Engine
To be taken as an example of others AXMEDIS Engine for Rules



6.2 Rule Scheduler

The Rule Scheduler is the application involved in the rules engines management. It plays the role of server in the distributed environment. It will be a multithread application and will be developed by using the wxWidgets ver. 2.4.2 library. This should allow having a multiplatform rule scheduler.

Referring to the picture, it is constituted of following parts:



AXMEDIS Distributed Engines

6.2.1 Scheduler Command Manager

The Scheduler Manager is the main application and provides the interface to the Scheduler GUI. The main provided functionalities are:

- Load/Save the configuration file.
- Initialisation of the scheduler
- Starting the scheduler
- Stopping the scheduler
- Exporting the list of rules and remote executors
- Browsing the list of jobs/rules and remote executors
- Performing backup of current status (list of rules)
- Performing the restore of the last list of rules
- Load/Install rules
- Providing commands for managing, controlling and monitoring the execution of rules and the activity of remote executors.
- Logs and reports management

The Scheduler Manager manages the configuration parameters stored in a “configuration.xml” file. The set of parameters contains information about:

- **Backup Time** - Backup interval for logging the set of submitted rule and tracing operations. It is expressed in minutes.
- **Time Out** - Time out on client activity. It is expressed in seconds.
- **Time Resolution** - Time Resolution of the scheduler. It is expressed in seconds.
- **Refresh Time** - Time Resolution for discovering new rule executors

- **Rules Path** - Rule Repository Path
- **Log Path** - Log Repository Path
- **Profile Path** - Executor Profile Repository Path
- **Backup Path** – The path where the scheduler periodically saves the current rules list.

The management of configuration parameters is done by using the AXMEDIS Configuration Manager.

6.2.2 Engine command and reporting (AXMEDIS Workflow Manager interaction)

It provides the communication and the interaction layer with the AXMEDIS Workflow Manager and to external applications such as the AXMEDIS Rules Editor. It is based on Web Service (GSOAP) and Workflow PlugIn. This module provides communication functionalities and allows managing:

- **Commands coming from the AXMEDIS Workflow Manager:**
  - *Run rule(AXRID, arguments, when)* – Ask for a running of a rule specified by means the AXRID, this command could override the arguments value of the rule by sending the xml descriptions about the *arguments* section(according to the XML schema) and could specify when the rule has to be run by overriding the *schedule*. If the command does not override the arguments the rule will be executed with the current arguments. If the command does not override the schedule, the rule will be executed immediately. Also all other combinations of overriding will be considered.
  - *Activate rule(AXRID)* – Switch the status of the rule specified by means the AXRID in the "ACTIVE" status
  - *Deactivate rule(AXRID)* - Switch the status of the rule specified by means the AXRID in the "INACTIVE" status
  - *Remove rule(AXRID)* – Remove the rule specified by means the AXRID from the Scheduler
  - *Pause rule*(AXRID) – Put the rule specified by means the AXRID in the "PAUSE" status
  - *Suspend rule*(AXRID) - Put the rule specified by means the AXRID in the "SUSPENDED" status for a specific time.
  - *Resume rule(AXRID)* – Resume the rule specified by means the AXRID
  - *Kill rule(AXRID)* – Stop the running of the rule specified by means the AXRID
  - *Install and activate rule (rule xml file)* – Install a new rule in the Scheduler by sending the XML file.
  - *Get list of rules* – Request the list of AXRIDs related to rules currently in the scheduler.
  - *Get logs(AXRID)* – Request the current action log generated by the Rule Scheduler. The action log will be pre-filtered to make available to the workflow an action log that is structured per each rule.
  - *Get rule status(AXRID)* – Request the current status of the rule specified by means the AXRID and the start time if rule is running or the next running time if it is not
  - *Get xml rule(AXRID)* - Request the XML file of the rule specified by means the AXRID

- **Reports, messages and files returned to the AXMEDIS Workflow Manager:**
  - Error notification (failure messages)
  - Complete rule notification
  - Log of the current activity
  - List of AXRIDs related to the currently scheduled rules
  - Xml file of Rule
  - Rule status (current status plus schedule information)

6.2.3 Communication with the AXMEDIS Rule Editor

- Commands coming from the AXMEDIS Rule Editor:
  - Get rules
  - Activate rule
  - Install rule in the scheduler (xml file transfer)
  - Run Rule

The communication between the AXCP Rule Editor and the Engine is Web Service based on the same protocol used by the Workflow Manager. This allows avoiding duplications.

6.2.4 Communication with other tools

The rulescheduler can be called by external tools that implement a client for calling Web Services based on the protocol used by the Workflow Manager. This solution allowed avoiding duplications. The Web Services are accessible if the external tool is registered in the configuration file of rule scheduler. The registration consists of providing the name of tool, the end point URI, the request ID according to the Workflow protocol. If it is not regesitered, the call is not served.
To receive back a notification, the external tool as to provide a server according to the Engine Channel WSDL definition and the provide the end point URI where the rulescheduler has to send back a response.
This allows having a return channel for creating the closure of a call to the GRID.

This approach is used to create the communication with the Rule Editor, calls coming from the AXMEDIS Rule Editor are:

- Get rules for browsing the GRID repository
- Activate rule for install and put in Active status a rule in the GRID
- Install rule in the scheduler (xml file transfer)
- Run Rule to ask the quick execution of a rule

6.2.5 Internal Scheduler

The internal scheduler is the manager of active rules. It has to detect, fire, launch and manage the execution of a rule. During its activity, the internal scheduler has to:

1. preserve the scheduled work from interruption of service (crash of the application) giving the possibility to restore the last status of activity
2. manage and update the list of rules to be scheduled and their status
3. manage and update the list of available rule executors
4. notify to the AXMEDIS Workflow Manager messages due to:
   - errors during the phase of rule association with an executor
   - errors due to the launching phase
   - errors during the rule execution on remote executor.
   - errors due to the time out deadline missing (the executor did not respond to request)

To this end, the functionalities provided by the rule scheduler are:

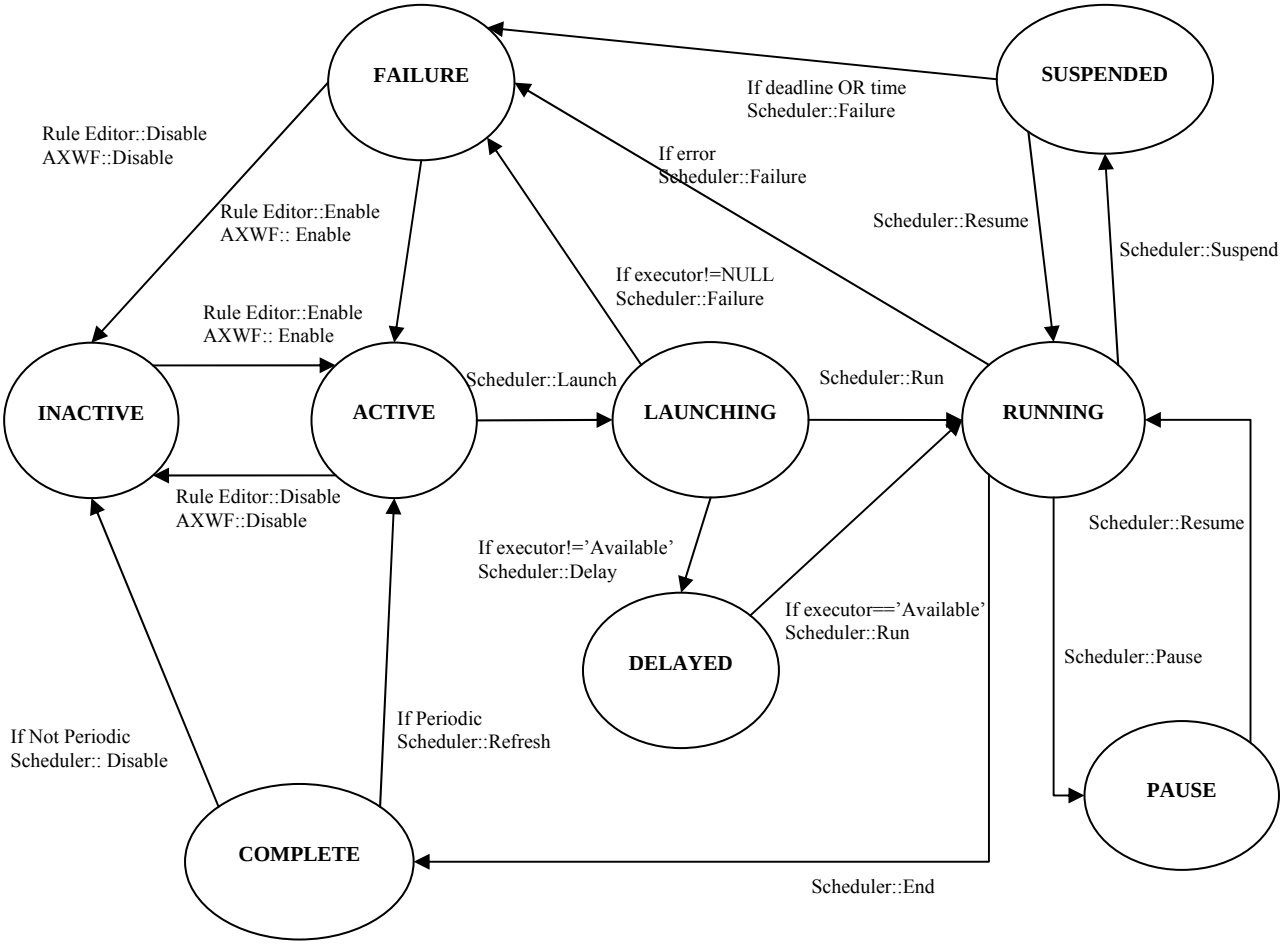
- Select from the internal scheduled rules the rule that matches conditions for the execution. This is performed by:
  - checking the execution time and date
  - receiving an immediate run command from the AXMEDIS Workflow Manager
- Modify and set the time resolution for the control of rules execution
- Add a new submitted rule in the list of jobs
  - Loading the corresponding rule xml file from the repository directory
  - Extracting the metadata for scheduling
  - Generating and assigning a Job Id to the rule
- Remove a rule from the list of jobs
- Run a rule on demand
- Reschedule a rule (by overriding the schedule information)
- Override rule arguments (by replacing the current arguments)
- Check expiration conditions of a rule
- Provide the list of jobs/rules
- Update firing conditions of a periodic rule
- Browse the list of jobs/rules
- Modify the status of rules
- Remove an executor from the list of executors
- Provide the list of executors

- Browsing the list of executors
- Save periodically on disk a backup copy of the list of jobs
- Restore the last status by loading the backup copy of the list of jobs
- Route messages coming from the dispatcher and the remote executors to the AXMEDIS Workflow Manager

Rule Life Cycle

The life cycle of rule represents the evolution of a rule from the activation to its completion. The evolution is strictly linked to the activities performed by the scheduler, the dispatcher and the executor and it can be described by means of a status attribute. The status of a rule assumes the following values:

1. **Active** – The rule is waiting for the execution
2. **Inactive** – The rule is not executed
3. **Failure** – An error occurred during the execution or control of rule. The execution is blocked and the executor is released.
4. **Launching** – The rule is associated with a remote executor.
5. **Delayed** – The launch of rule has been delayed. Available executors are temporary busy with another rule.
6. **Running** – The rule is running on a remote executor
7. **Pause** – The run of rule has been stopped temporary
8. **Suspended** – The run of rule is suspended by defining a temporal interval. The time unit is second.
9. **Complete** – The run is finished



State Diagram for an AXCP rule

The general evolution of the rule status is depicted in the State Diagram representation, whereas the description of each transition is reported in the following.

Complete to Active Transition – The transition from COMPLETE to ACTIVE status is performed if the rule has to be executed periodically. In this case, the rule is re-submitted to the scheduler and its run-conditions are updated on the basis of the specified period.

Complete to Inactive Transition – The transition from COMPLETE to INACTIVE status is performed when the rule has to be executed once. In this case, the rule is ready to be removed from the scheduler or to be modified by means of the Rule Editor or to be run on demand.

Active to Launching – The transition from ACTIVE to LAUNCHING status is performed when the scheduler fires a rule. The rule has to be associated with a remote executor.

Launching to Running Transition – The transition from LAUNCHING to RUNNING status is performed when a rule is running on the remote executor.

Launching to Delayed Transition – The transition from LAUNCHING to DELAYED status is performed when the executor that could run the rule is busy. The rule is placed temporally in a delay queue to be run when the executor will be ready.

Delayed to Running Transition – The transition from DELAYED to RUNNING status is performed when the remote executor is ready to run the rule.

Launching To Failure Transition – The transition is performed when during the rule check operation, the rule profile does not match any available executor profiles.

Running to Failure Transition – The transition from RUNNING to FAILURE status is performed if the following conditions occur:

- when during the execution of rule the corresponding rule executor sends a run-time error message.
- when during the execution of the rule, the executor does not respond to sever call (Time out).

Running to Complete Transition – The transition from RUNNING to COMPLETE status is performed when the execution of rule is successfully completed.

Running to Pause Transition – The transition from RUNNING to PAUSE status is performed when a pause rule request comes from the Scheduler. The status of PAUSE can be conditioned by deadline condition.

Pause to Running Transition – The transition from PAUSE to RUNNING status is performed when a resume rule request comes from the Scheduler.

Failure to Inactive Transition – The transition from FAILURE to INACTIVE status is performed when a disable rule request comes from the AXMEDIS Workflow Manager or from the Rule Editor.

Failure to Active Transition – The transition from FAILURE to ACTIVE status is performed when a disable rule request comes from the AXMEDIS Workflow Manager or from the Rule Editor.

Suspended to Failure Transition – The transition from SUSPENDED to FAILURE status is performed when the suspension misses the deadline condition.

Running to Suspended Transition – The transition from RUNNING to SUSPENDED status is performed when a rule suspension request comes from the Scheduler. The status of SUSPENDED is conditioned by a deadline condition (e.g. a temporal interval).

Suspended to Running Transition – The transition from SUSPENDED to RUNNING status is performed when a resume rule request comes from the Scheduler. Since the status of SUSPENDED can be conditioned by deadline condition, a rule is automatically resumed by the scheduler when the deadline condition is matched.

Jobs as Rules
From the scheduler point of view, a rule is a job to be executed. Each job is described by using the metadata contained in the *Header*, *Schedule* and *Definition* section of the rule.
Jobs are organised in a table, called **Table of Jobs.** Such table is monitored by the internal scheduler periodically to detect the rule to be fired. A periodic backup of the table is performed to guarantee a certain degree of reliability in case of critical problems that could require to restart the scheduler. The backup copy will permit to restore the last status of the scheduler.

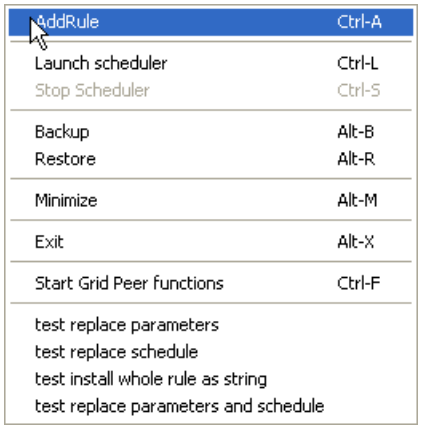
Job	
Attributes	**Description**
Rule Name	The name of rule. It comes from the corresponding data in the Header section of the rule
Rule Version	The version of rule. It comes from the corresponding data in the Header section of the rule
Rule Type	The type of rule. It comes from the corresponding data in the Header section of the rule
Start Time	The start time of execution. It comes from the corresponding data in the Schedule section of the rule
Start Date	The start date of execution. It comes from the corresponding data in the Schedule section of the rule
Periodicity	The name of rule. It comes from the corresponding data in the Schedule section of the rule
Expiration Time	The expiration time of job life. It comes from the corresponding data in the Schedule section of the rule
Expiration Date	The expiration date of job life. It comes from the corresponding data in the Schedule section of the rule
Executor ID	It is the identifier of the current executor associated with the rule by the Dispatcher
N° of Run	Number of times that the rule has been run
Job ID	Identifier associated with the rule by the scheduler
Profile	List of dependencies (AXMEDIS Plug In, external tools, etc…). It comes from the Uses data in the Definition section of the rule
URL	The complete path of the rule xml file.

6.2.6 Rule Scheduler User Interface
The Scheduler GUI will be the main window that allows the interaction with the Scheduler. It is constituted of:

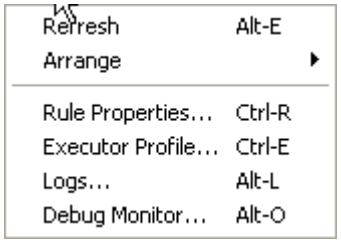
1. A menu bar
2. Two main areas where the list of rules and the list of remote executors are displayed.
3. A status bar where the current clock and the current date are displayed.

Menu bar – It provides the access to the following set of functions:

1. **Program**
    a. *Add Rul*e – It allows to load rules and install into the scheduler
    b. *Launch scheduler* - Start the scheduler activity.
    c. *Stop scheduler* - Stop the scheduler activity.
    d. *Backup* - Backup Copy of the last jobs list.
    e. *Restore* – Restore a backup copy
    f. *Minimize* - It reduces at icon on the taskbar.
    g. *Exit* - Close the application.
    h. *Start Grid Peer Functions* – It starts the components for P2P network access



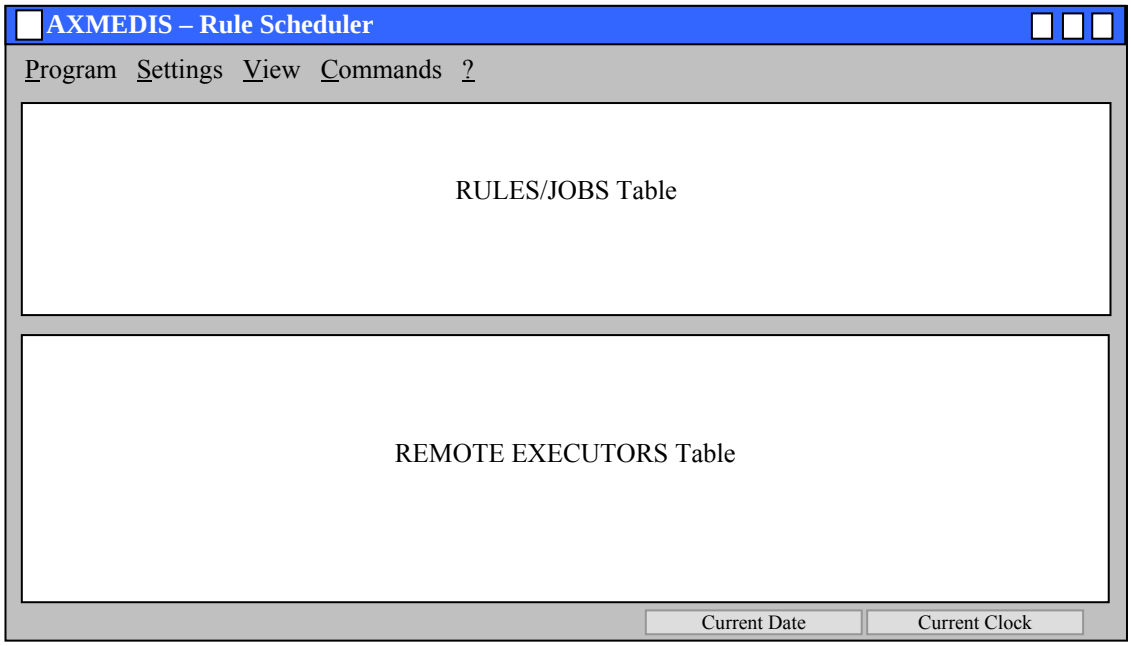
2. **Settings**
   a. *Preferences* - Open an editable dialog with the set of configuration parameters.
3. **View**
   a. *Refresh* – Update the list of jobs and list of remote executors.
   b. *Arrange* – Repainting modes of tables in the main frame
      i. *Top* – It shows only the top table (Table of rules)
      ii. *Bottom* – It shows only the bottom table (Table of executors)
      iii. *Vertical* – It shows tables vertically
      iv. *Horizontal* – It shows tables horizontally
   c. *Rule Properties…* - Open a Rule Properties dialog.
   d. *Executor Profile…*- Open an Executor Profile dialog.
   e. *Logs…*- Open a dialog to show the list of log messages
   f. *Debug Monitor…*A dialog for debug purpose

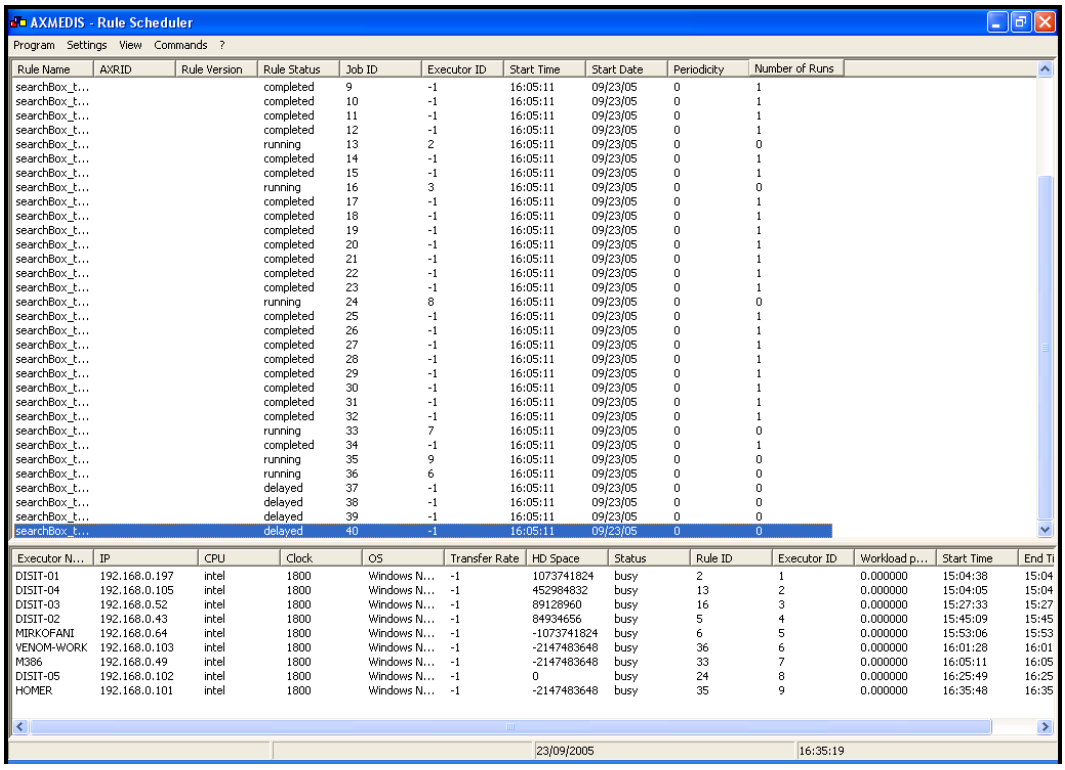


4. **Commands**
   a. *Enable Rule* - Put in the "ACTIVE" status the current selected inactive rule.
   b. *Disable Rule* - Put in the "INACTIVE" status the current selected active rule.
   c. *Kill Rule* - Kill the current execution of the current selected rule.
   d. *Pause Rule* - Put in pause the execution the current selected rule.
   e. *Resume Rule* - Resume the execution of the current selected rule.
   f. *Suspend Rule…* - Open a dialog to edit the temporal interval for rule resuming and then suspend the current selected rule.
   g. *Remove Rule* – Remove the rule from the list of rules

5. **?**
   a. *Help* - Open the On Line help.
   b. *About* - Open a dialog with credits.

All this functionalities are also accessible by means shortcuts.





Rules/Jobs Table - It is the area where scheduled rules are displayed. It is a list control constituted of a set of columns where the following list of metadata are displayed:

- *Rule name* –the name of the rule
- *Rule version* –the version of rule
- *Rule status* –the current status of rule
- *Rule ID* –the identifier of rule
- *Executor ID* –the identifier of the executor associated with rule
- *Start Time* –the time to fire the rule
- *Start Date* –the date to fire the rule

- *Periodicity* –the periodic attribute
- *N° Runs* –the number of time the rule was fired.

Name	Version	Status	ID	Executor ID	Start Date	Start Time	Periodicity	N° Runs

The following functionalitiesare provided by means a contextual popup menu:

- Ordering rules alphabetically by name
- Ordering rules by start running time
- Ordering rules by ID

Remote Executors Table - It is the area where remote executors are displayed. It is a list control constituted of a set of columns where the following list of metadata is displayed:

- *Name* - Computer Name
- *IP* - IP address
- *CPU* - CPU & Clock
- *OS* - OS & Version
- *Ping* – The network capabilities in term of transmission time.
- *HD Space* – The space available on the disk of the executor
- *Status* – The status of the executor
- *Rule ID* – The ID of the running rule
- *Executor ID* – The Id of the executor assigned by the scheduler
- *Start Time* – At what time the run is started.

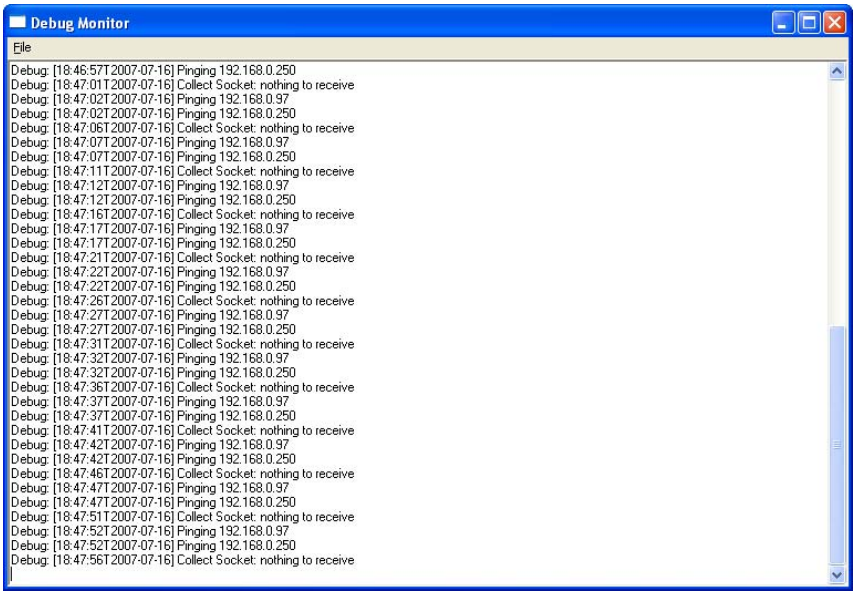
Name	IP	CPU	OS	Ping	HD Space	Status	ID	RuleID	Start Time

The following functionalities are provided by means a contextual popup menu:

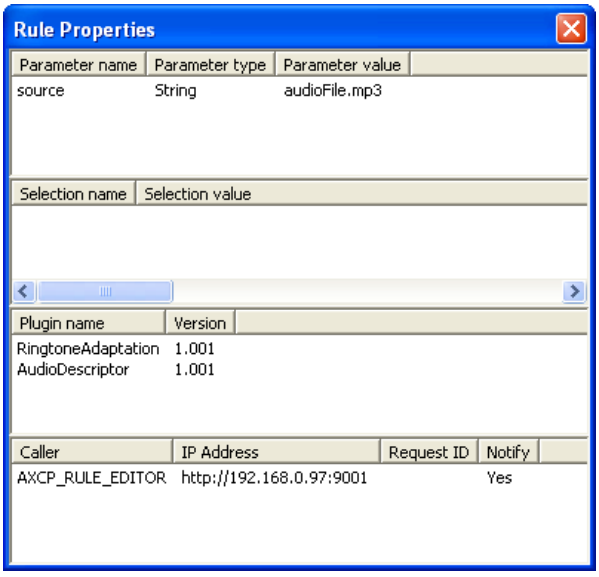
- Ordering executors alphabetically by computer name
- Ordering executors by ID

Auxiliary dialogs
The Scheduler GUI is supported by the following set of dialogs:

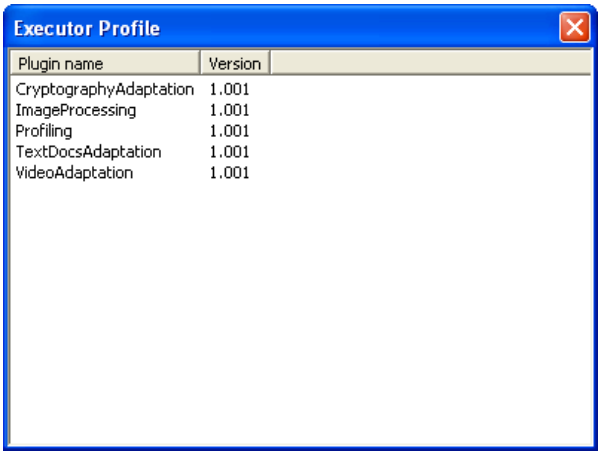
Debug Monitor Dialog – This dialog shows the current activity of the Scheduler providing both runtime messages and incoming messages sent by grid node. It allows monitoring at runtime the internal activity of the rule scheduler and grid.



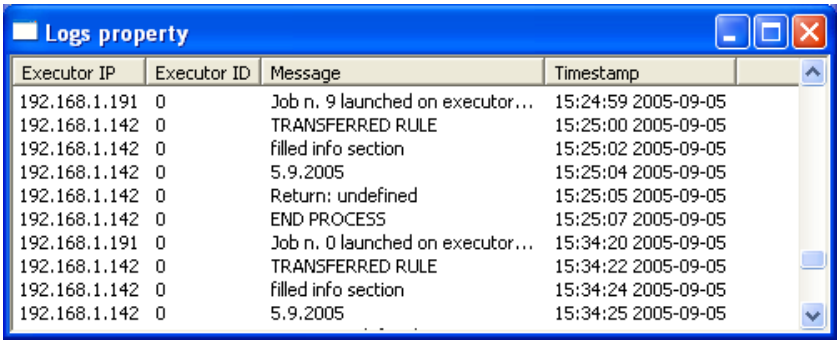
Rule Properties Dialog – It is an not editable no modal dialog where the properties of the selected rule are displayed. Some of these properties are extracted from the XML file associated with rule.



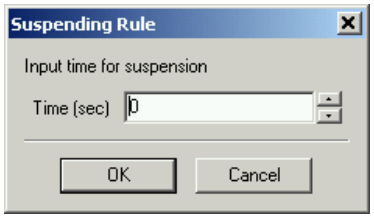
Executor Profile Dialog - It is a not editable no modal dialog where the properties of the selected executor are displayed. Some of these properties are extracted from the executor profile.



Logs Dialog – This dialog allows viewing the logs of scheduler activity. The dialog can be refreshed by pressing the F5 key.



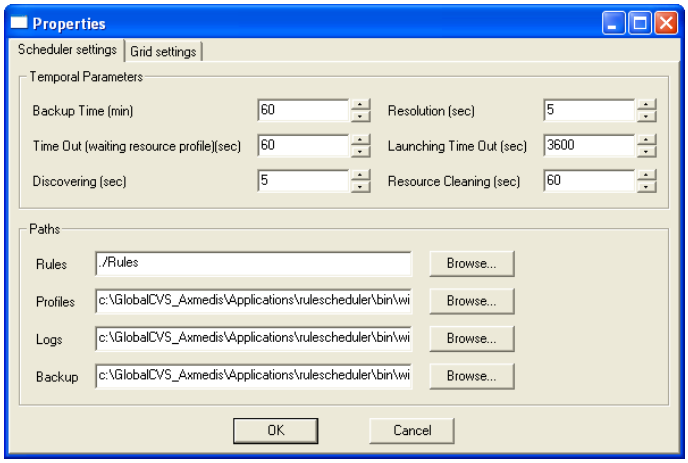
Suspend Rule Dialog – It is an editable no modal dialog where the user puts the time for the suspension.



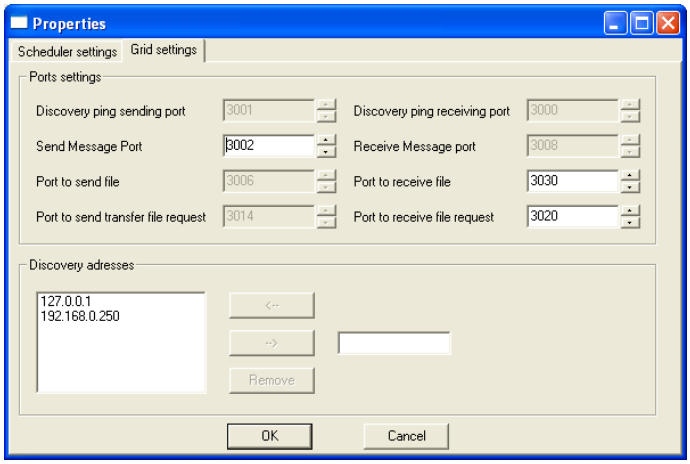
Preferences Dialog – It is a tabbed dialog that allows editing settings parameters regarding the scheduler activity (*Scheduler settings*) and the GRID support (*GRID settings*).

Scheduler settings – It consists of a set of configuration parameters contains settings about:

- **Backup Time** - Backup interval for logging the set of submitted rule and tracing operations. It is expressed in minutes.
- **Time Out** - Time out on client activity. It is expressed in seconds.
- **Time Resolution** - Time Resolution of the scheduler. It is expressed in seconds.
- **Refresh Time** - Time Resolution for discovering new rule executors
- **Rules Path** - Rule Repository Path
- **Log Path** - Log Repository Path
- **Profile Path** - Executor Profile Repository Path
- **Backup Path** – The path where the scheduler periodically saves the current rules list.
- **Resource Cleaning** – Specify the periodicity for cleaning the GUI by removing diaplsyed completed rules.



Grid settings – It provides a set of settings to setup the communication support. It allows to define the number of ports to use when receiving file, messages, sending files, responding to the discovering request. It allows also to define IPs of LANs to use when the scheduler performs the discovering of peers.



6.2.7 Dispatcher of the Rule Scheduler

The main role of the dispatcher is to:

- associate a remote executor with the rule to be run
- engage the selected remote executor
- launch the execution of rule on the remote executor
- monitoring the status of remote executors
- managing the possible errors messages and notifications coming from executors
- creating logs and tracing the activity of each remote executor involved in the execution of a rule
- discovering new remote executor
- requesting and receiving the profile from remote executors

Remote Executors

A remote executor is the virtual image on the scheduler side of a real machine equipped with the rule executor. Knowing the availability and capabilities of a remote executor is mandatory to identify the machine that will execute the rule. To this end, the association of executors with rules is based on the list of available remote executors (computer) and their profiles. Such list is persistent and it is managed at run-time. Each remote executor belonging to the list is described by means of internal attributes (managed by the scheduler) and a profile that is sent by each real executor during the discovery and refreshing phase. The profile contains a set of metadata that describes the capabilities of the remote executor (see Section 6.9.6).

Executor	
Attributes	**Description**
Current Status	It provides the status of the executor
Executor ID	Identifier of the executor
Profile	It is the set of information related to : 1. Identity of the executor (computer name, IP address, location, etc…) 2. Computational capabilities : (CPU, RAM, Clock, etc…) 3. Functionalities that the executor provides: • AXMEDIS Plug-In installed (For each plug in the name and version are provided). • External tools Plug-In installed For each plug in the name and version are provided).

To realise that, the dispatcher is divided in four main components.

Resource Controller – It periodically controls and refreshes the availability of remote executors in the network of AXMEDIS factory. To this end, it performs the following activities:

- discovering new remote executors
- requesting profiles of remote executors
- refreshing and managing the list of available remote executors (adding a new discovered executor, remove an executor from the list, loading the profile)
- generating and assigning an unique Executor ID (process Id) with the remote executor
- initialising the remote executor (by sending the Executor ID, the port number to use for sending messages to the Rule Monitor)
- managing and updating the status of each remote executor.
- generating an error in case of time out or deadline missing (the remote executor did not respond to a request within a time interval specified by a time out)

Optimizer – It receives rules to be launched. They are put in an internal queue that includes rules to be associated with a remote executor. The choice of an executor is performed by checking the rule profile with the best profile among available remote executors. If all available remote executors do not match the profile of rule, the association fails, the status of rule is set to "Failure" and an error message is generated and sent to the internal scheduler. If all remote executors matching the profile of the current rule are running different rules, the launch of the current rule is delayed and the rule is maintained in the internal queue of rules waiting for an available executor.
Future version of the optimizer will include optimisation algorithm based on Deadline monotonic algorithm that will improve the scheduler capability.

Rule Launcher – The rule launcher:

1. sends commands generated by the scheduler (kill, pause, run, resume) to remote executors
2. engages the remote executor associated with the rule and launch the execution of rule. If the engaging fails an launch error is generated.

To this end, the rule launcher provides functionalities for:

- communicating with the remote executor (by sending and receiving messages and commands)
- transferring the rule to the remote executor
- communicating errors messages to the internal scheduler if the launch phase fails
- tracing the commands and controls sending to remote executors by updating an activity log file

Rule Monitor – It monitors persistently the execution of rules by:

- listening to messages and notifications coming from remote executors
- interpreting messages

- managing the log file and trace the activity of each rule executor (by reporting all messages and notifications in input)
- communicating the status of a rule to the internal scheduler for updating
- routing possible errors messages and notifications to the internal scheduler

To this end, the rule monitor provides functionalities for:

- communicating with the remote executor (by receiving messages and notifications)
- parsing messages of executors
- providing and managing a queue of input messages
- updating log files associated with each remote executor for each message or notification received
- routing errors and messages to the internal scheduler coming from executors

6.2.8 Grid Peer Interface

It is the interface to the Grid Peer. It provides functionalities for:

- Sending message to a peer
- File transfer to a peer
- Discovering peers
- Engaging/Launching a peer
- Managing messages coming from other peers

6.2.9 Grid Peer

It provides the support for the distributed system management. It is based on TCP/UDP socket and is constituted of the following components:

- **Peer Explorer** – It provides functionalities and support for querying the presence of other peers. It is based on UPD broadcast messages.
- **Peer Communicator** – It provides functionalities and support for communicating with available peers (already discovered). It is based on TCP connection.
- **Peer File Transfer** – It provides functionalities and support for transferring file to a selected peer. It is based on TCP connection.
- **Peer Event Consumer** – It provides functionalities and support for handling events of communication, file transfer and discovering.

These componentsare used singularly and independently. It was developed in C++ with STL and WindowsSocket library.

6.2.10 Structure of messages exchanged between Scheduler and Remote Executor

Messages exchanged between the Scheduler and the Remote Executor can be different types and grouped in two set of messages: (i) from Scheduler to Rule Executor and (ii) from Rule Executor to Scheduler.

Messages from Scheduler to Rule Executor:

1. **Command** – the message is a specific command

Messages from Rule Executor to Scheduler:

2. **Notification** – the message is a notification
3. **Error** – the message reports an error
4. **Response** – the message is a response to a request or a command

The main idea is to have a common message structure that allows covering all these types. In addition, to guarantee a fast delivery on the network, messages are light. To this end, they are based on a formatted text and structured according to the following EBNF formalisation:

<message> := <Sender ID>'#'<Type_Msg>

<Sender ID> := <string>
<Type_Msg> := <CMD_MSG> | <REQ_MSG> | <NOTIFY_MSG> | <ERR_MSG> | <RESP_MSG>

<CMD_MSG> := 'COMMAND#'<ID_MSG>'#'<command>
<command> := RUN | KILL | PAUSE | RESUME | GET <request> | SET <attribute> <value> | TEST <attribute> <value>
<request> := PROFILE | STATUS | ID
<attribute> := ID | …
<value> := <string>

<NOTIFY_MSG> := 'NOTIFICATION#'<what notified>
<what notified> := 'END PROCESS' | <msg>
<msg> := 'MSG' <string>

<ERR_MSG> := 'ERROR#'<error from>'#'<error description>
<error from> := 'RULE' | 'EXECUTOR'
<error description> := <error code> | <string>

<RESP_MSG> := 'RESPONSE#'<to msg>'#'<response argument>
<response argument> := <status> | <executor ID> | 'CMD OK'
<to msg> = <ID MSG>

<ID MSG> = <timestamp>

Where:
<timestamp>: it indicates the generation time of a message and allows indexing a message. It could be used as reference to link a response message to command messages and to monitor the activity of the rule executor.

<Sender ID>: it indicates the identifier of the sender. By default, the ID of the Scheduler is '0', whereas for all rule executors will be the Executor ID

<error code>: it reports the code of the error

Example 1:
The scheduler requests the profile to a rule executor by means the message:

0#COMMAND#12:00:00 pm#GET PROFILE

where:

- '0' is the sender ID associated with the scheduler (server).

Example 2:
The scheduler sends to the rule executor its Executor ID:

0#COMMAND #12:20:00 pm#SET ID 34

The scheduler requests the value of status to the executor identified by "34" by means of the message:

0#COMMAND #12:20:00 pm#GET STATUS

The rule executor "34" responds to the request by means of:

34#RESPONSE#12:20:00 pm#'value of status'

where:

- ‘34’ is the sender ID associated with the Executor ID of the rule executor.

Example 4:
The rule executor ‘34’ sends to the scheduler:

- a message generated by the rule:

34#NOTIFICATION#MSG ”AXMEDIS Database connection error”

- a run time error

34#ERROR#EXECUTOR#”Disk Full” or 34#ERROR#EXECUTOR#001

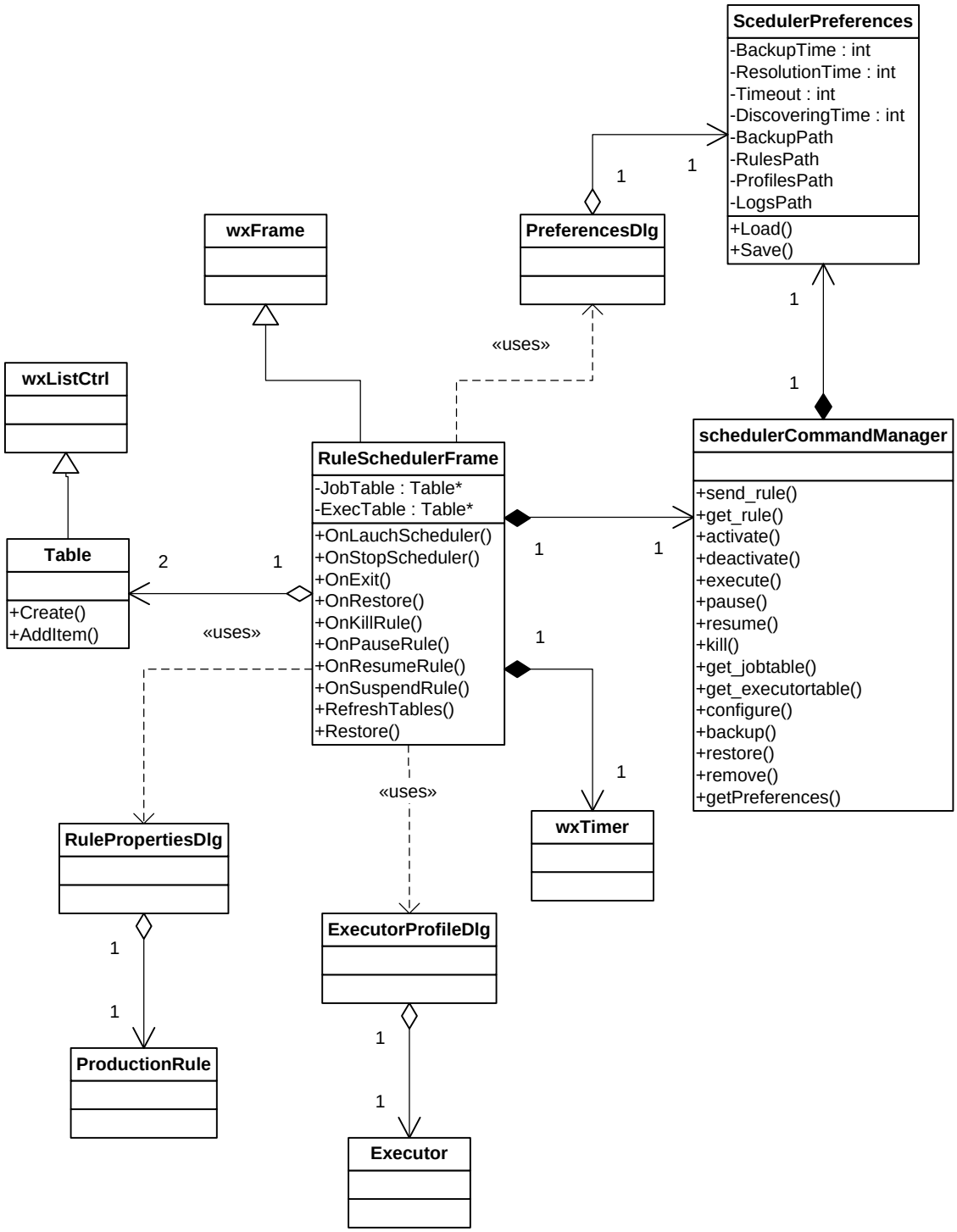
where ‘001’ could be for instance the error code associated with “Disk Full”

- an end process notification

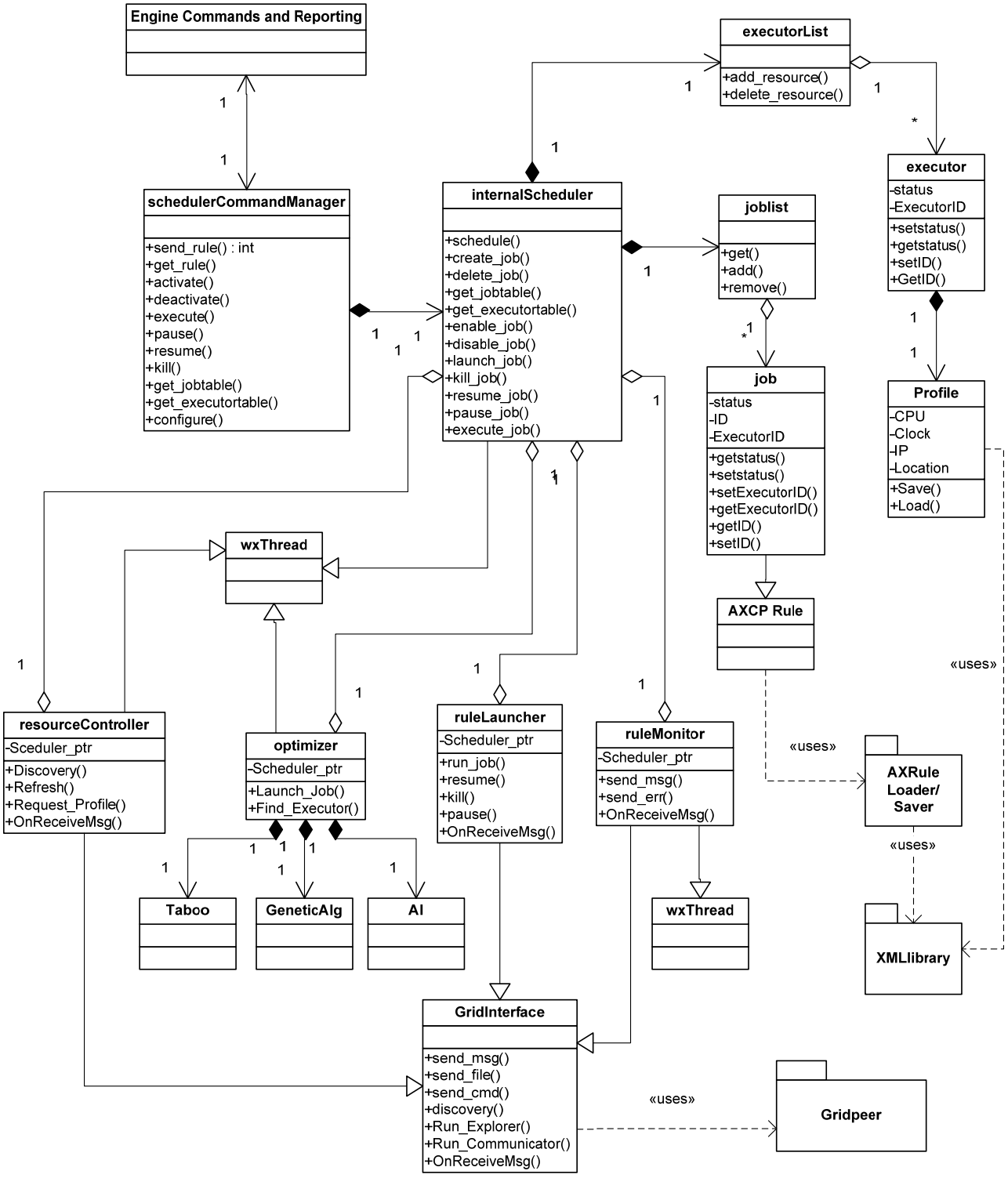
34#NOTIFICATION#END PROCESS

6.2.11 Rule Scheduler Class Diagram

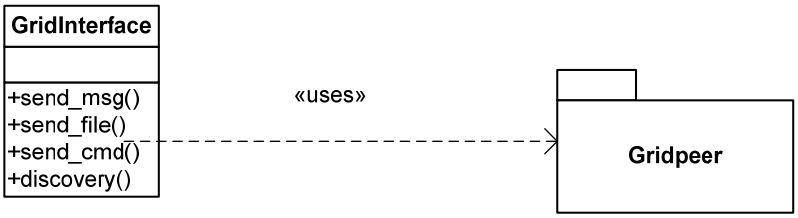
Class diagram of Scheduler GUI

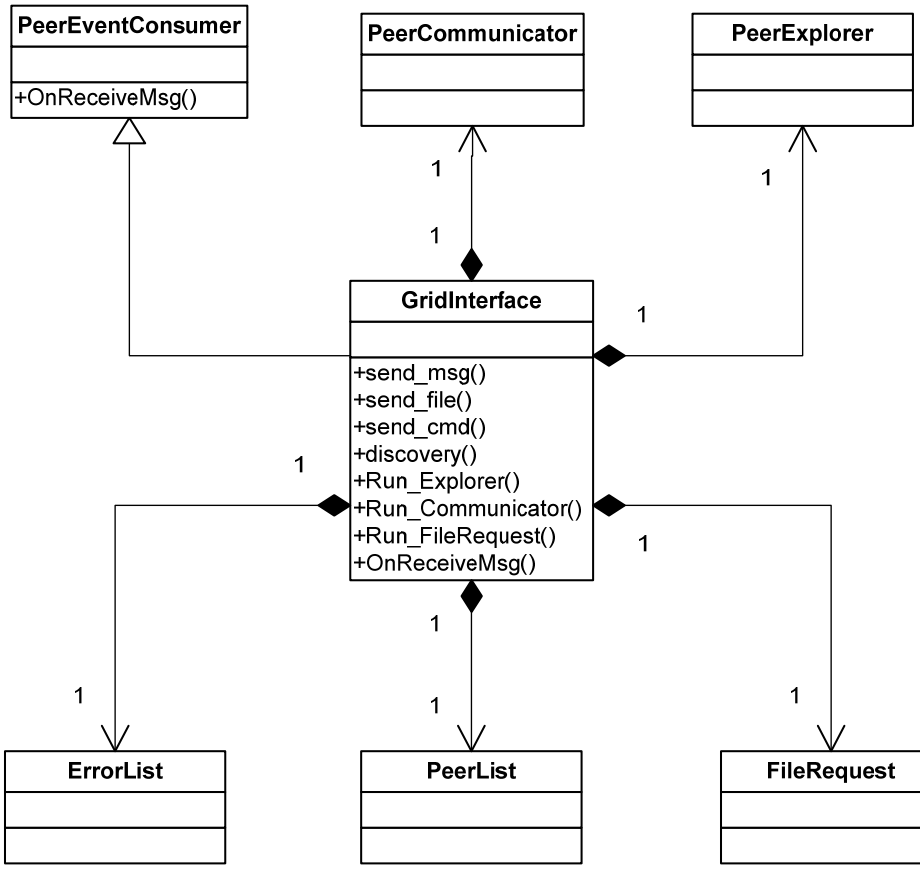


Class diagram of Internal Scheduler


Engine Commands and Reporting
executorList
+add_resource()
+delete_resource()
executor
-status
-ExecutorID
+setstatus()
+getstatus()
+setID()
+GetID()
schedulerCommandManager
+send_rule() : int
+get_rule()
+activate()
+deactivate()
+execute()
+pause()
+resume()
+kill()
+get_jobtable()
+get_executortable()
+configure()
internalScheduler
+schedule()
+create_job()
+delete_job()
+get_jobtable()
+get_executortable()
+enable_job()
+disable_job()
+launch_job()
+kill_job()
+resume_job()
+pause_job()
+execute_job()
joblist
+get()
+add()
+remove()
job
-status
-ID
-ExecutorID
+getstatus()
+setstatus()
+setExecutorID()
+getExecutorID()
+getID()
+setID()
Profile
-CPU
-Clock
-IP
-Location
+Save()
+Load()
wxThread
AXCP Rule
«uses»
resourceController
-Sceduler_ptr
+Discovery()
+Refresh()
+Request_Profile()
+OnReceiveMsg()
optimizer
-Scheduler_ptr
+Launch_Job()
+Find_Executor()
ruleLauncher
-Scheduler_ptr
+run_job()
+resume()
+kill()
+pause()
+OnReceiveMsg()
ruleMonitor
-Scheduler_ptr
+send_msg()
+send_err()
+OnReceiveMsg()
«uses»
AXRule Loader/ Saver
«uses»
Taboo
GeneticAlg
AI
wxThread
XMLlibrary
GridInterface
+send_msg()
+send_file()
+send_cmd()
+discovery()
+Run_Explorer()
+Run_Communicator()
+OnReceiveMsg()
«uses»
Gridpeer

Class diagram of the GridInterface class





6.3 Configuration Parameters

In this section the set of parameters regarding the configuration of the rule □cheduler are listed. Such parameters are grouped into modules as reported below:

6.3.1 AXMEDIS Rule Scheduler Frame

Config parameter	Possible values
FRAME_SIZE	it is the frame size information
FRAME_POSITION	it is the frame position information
XML_XSD_PATH	It is the directory where xml schema (XSD files) are stored

6.3.2 AXMEDIS Rule Scheduler Settings

Config parameter	Possible values
BACKUP_TIME	It is the interval of Backup
RESOLUTION	It is the resolution in terms of time

TIME_OUT	It is the value of time out for waiting in the communication
DISCOVERING	It the discovering time
LAUNCHING_TIME_OUT	It is the maximum interval time for launching a rule
RULES_PATH	it is the directory where the rule will be saved
PROFILES_PATH	It is the directory where profiles of executors are stored
BACKUP_PATH	It is the directory where the backup file is saved
LOGS_PATH	It is the directory where logs are stored
TIME_OUT_CLEAN	It is the interval for cleaning the GUI by removing Completed rules

6.3.3 AXMEDIS_GRID_SUPPORT_SETTINGS

Config parameter	Possible values
SERV_PORT	It is the value of the port used by the Grid Interface
SERV_PORT2	It is the value of the port used by the Grid Interface
SERV_PORT3	It is the value of the port used by the Grid Interface
SERV_PORT4	It is the value of the port used by the Grid Interface
SERV_PORT5	It is the value of the port used by the Grid Interface
DEFAULT_NET	It is the list of IP address

6.3.4 AXMEDIS Plugin Manager

Config parameter	Possible values
PLUGINS_PATH	it is the frame size information

6.3.5 WORKFLOW

Config parameter	Possible values
workflowUrl	it is the URL for workflow plugin
gatewayUrl	it is the gatewayURL for workflow plugin
WorkflowPluginId	It is the indentificator for the workflow plugin (dll)

In the following an example of configuration file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Configuration xmlns="http://www.axmedis.org/configuration">

  <Module category="" id="AXMEDIS_RULE_SCHEDULER_FRAME">
    <Parameter name="FRAME_SIZE" type="string">738:1024</Parameter>
    <Parameter name="FRAME_POSITION" type="string">0:0</Parameter>
    <Parameter name="XML_XSD_PATH"
type="string">C:\Axmedis\axmedis\Applications\rulescheduler\bin\win32</Parameter
>
  </Module>

  <Module category="" id="AXMEDIS_RULE_SCHEDULER_SETTINGS">
    <Parameter name="BACKUP_TIME" type="string">30</Parameter>
    <Parameter name="RESOLUTION" type="string">5</Parameter>
    <Parameter name="TIME_OUT" type="string">60</Parameter>
    <Parameter name="DISCOVERING" type="string">2</Parameter>
    <Parameter name="LAUNCHING_TIME_OUT" type="string">3600</Parameter>
    <Parameter name="RULES_PATH" type="string">.\Rules</Parameter>
    <Parameter name="PROFILES_PATH" type="string">.\Profiles</Parameter>
    <Parameter name="BACKUP_PATH" type="string">.\Backup</Parameter>
    <Parameter name="LOGS_PATH" type="string">.\Logs</Parameter>
    <Parameter name="TIME_OUT_CLEAN" type="string">30</Parameter>
  </Module>
```

```
  <Module category="" id="AXMEDIS_GRID_SUPPORT_SETTINGS">
    <Parameter name="SERV_PORT" type="string">3000</Parameter>
    <Parameter name="SERV_PORT2" type="string">3001</Parameter>
    <Parameter name="SERV_PORT3" type="string">3002</Parameter>
    <Parameter name="SERV_PORT5" type="string">3014</Parameter>
    <Parameter name="SERV_PORT4" type="string">3006</Parameter>
    <Parameter name="DEFAULT_NET"
type="string">192.168.1.253:192.168.1.252:192.168.1.149:192.168.1.31:192.168.1.2
51:192.168.1.12</Parameter>
  </Module>

  <Module category="" id="AXMEDIS_PLUGIN_MANAGER">
    <Parameter name="PLUGINS_PATH">./Plugin</Parameter>
  </Module>

  <Module category="" id="WORKFLOW">
    <Parameter name="workflowUrl"
type="string">http://192.168.1.118:8080/Prove_WF</Parameter>
    <Parameter name="gatewayUrl"
type="string">http://192.168.1.118:8000/responseGateway/engineChannel/engineChan
nel.asmx</Parameter>
   <Parameter name="WorkflowPluginId" type="string"></Parameter>
  </Module>
</Configuration>
```

6.4 Technical and Installation information

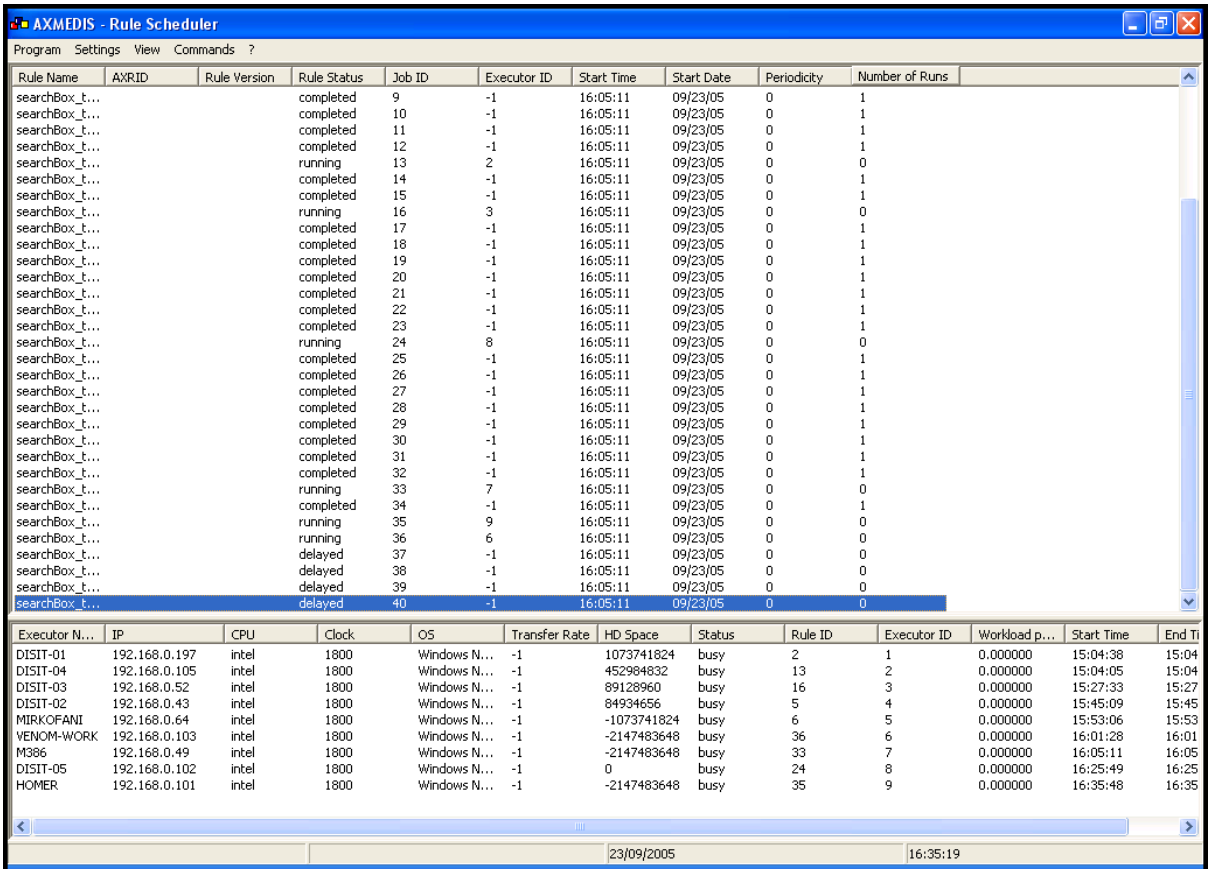
References to other major components needed	The AXCP Grid Node works in conjuction with the AXCP Rule Scheduler.
Problems not solved	The current version is not equipped with a system to limit the request of CPU. So it could happen that the run of a rule requires 100% of CPU. This will be solved in the next version.
Configuration and execution context	The Application provides a configuration file.

6.5 Draft User Manual

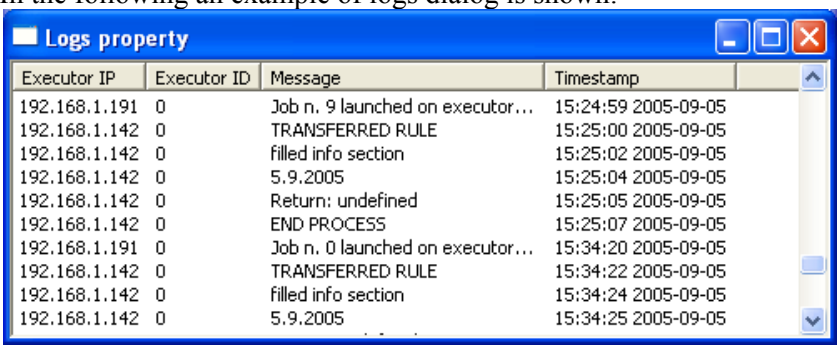
When the user is happy with his AXCP Rule and validated it with a quick and/or full trial, the final option is to activate the AXCP Rule in the AXCP Rule Engine. In the current version of the prototype the user has to install manually the AXCP Rules in the Scheduler by selecting "*Add rule*" in Program menu.
The command opens a load Dialog for single or multiple selection of AXCP Rules.
To start the scheduler, the Start Scheduler command in the "Program" menu has to be called, successively the "Start Grid Functions" allows starting the P2P interface. The scheduler is ready to discover possible AXCP Rule executor in the network. The discovered executors will be displayed in the Executor List area.
The scheduler will process information of rules and at the specified times will distribute rules to a rule executor. During the running of the rule, the user can also stop it from the Scheduler by selecting "*Kill Rule*" in the Command menu or other commands to control the execution of rule. The logs is accessible in real time by calling the "Logs" command in the "View" menu. By pressing the F5 key the logs window is refreshed.

6.6 Examples of usage

The following snapshot shows the main window at runtime:



In the following an example of logs dialog is shown:



6.7 Integration and compilation issues

It is a stand alone window application.

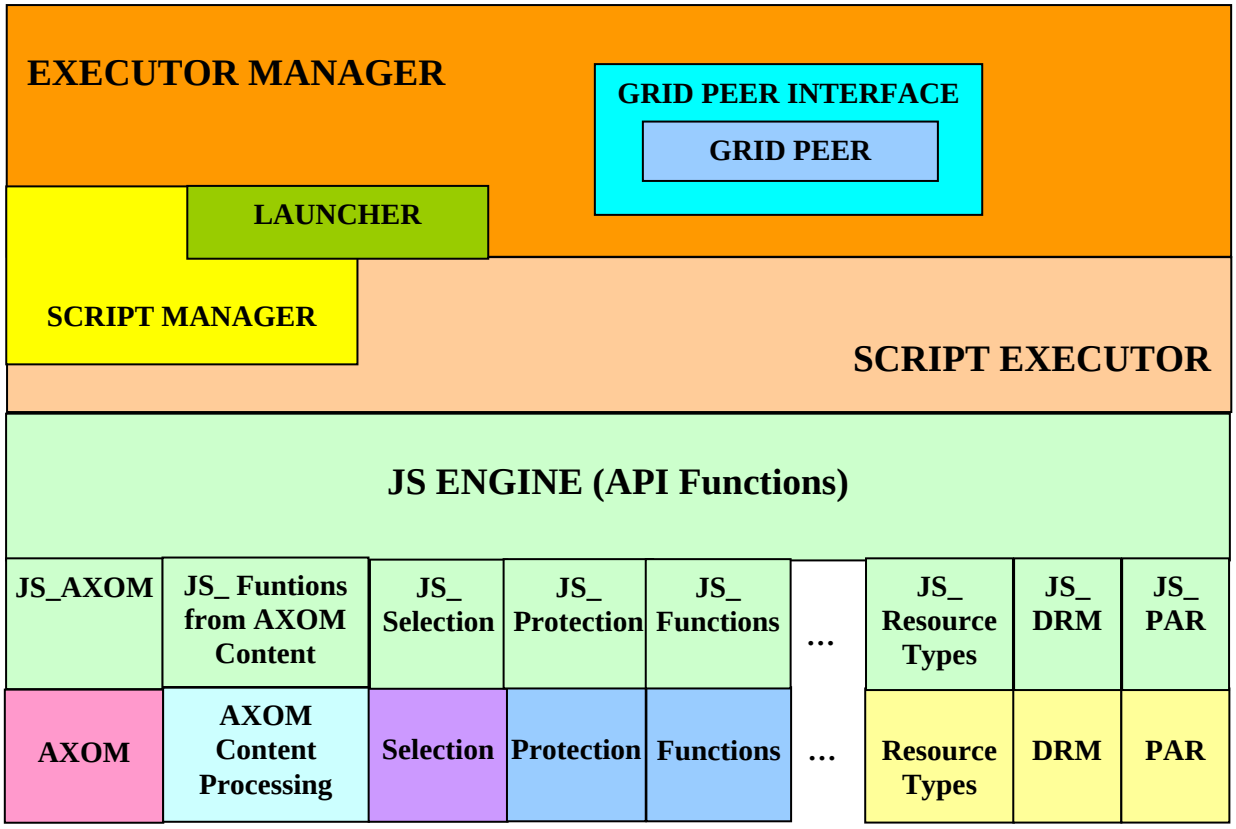
6.8 Errors reported and that may occur

Not available at this step.

Error code	Description and rationales

6.9 Rule Executor (AXCP GRID Node)

The rule executor is an application running on a remote computer. It is a computational unit in the distributed rule engines environment. It is based on Javascript engine and executes JavaScript code. To this end it hosts the SpiderMonkey Javascript Ending realised by Mozilla. The main architecture of the Rule Executor is depicted in the following picture:



AXMEDIS Rule Executor

The main components are:

- **Grid Peer Interface** – the communication support with the AXCP rule scheduler
- **Rule Executor Manager** – the command interface of the engine
- **Script Executor** – It hosts the SpiderMonkey Javascript Engine (called JS Engine)

6.9.1 Executor Manager

The Rule Executor Manager is the main program and the interface between the script executor and the scheduler. It hosts the Grid Peer Interface for the communication in the distribute environment based on the technology of the Grid Peer on the scheduler side. The interface allows:

1. receiving commands, messages, requests and files from the Scheduler
2. sending messages, notifications and files to the Scheduler
3. being discovered by the Scheduler during the discovering phase.

The main activity is to:

- generate the profile of the executor to send to the Scheduler
- receive the rule file from the Scheduler
- load the rule in the executor
- manage the launch of the rule execution by means the *Launcher*
- provide to the *Script Executor* the support of communication with the Scheduler.

- notify errors, status and the end of execution to the Scheduler.

To realize that, the Rule Executor Manager provides functionalities for:

- Routing messages produced by internal components to the scheduler
- Receiving control messages and commands from the Scheduler
- Parsing and executing commands coming from the scheduler such as:
  - Launch the execution of rule
  - Kill the execution of rule
  - Pause the execution of rule
  - Resume the execution of rule
  - Request profile
  - Request status
- File Transferring to:
  - send the profile of the Rule Executor
  - receive the rule to be executed
- Sending messages and notification to the Scheduler
- Creating the profile of the executor according to the XML schema
- Managing the status of the Executor

Status of the Rule Executor
The Rule Executor status describes the activity of the rule engine. If it is available for running a rule the status value will be "READY", otherwise if it is working with a rule the value will be "RUNNING". In event of errors that could break the execution, if they could be managed by software, the executor notifies them to the Scheduler stopping definitively the current execution and resetting the status to the "Ready" value.

6.9.2 Launcher

The role of the Launcher is to start the execution of the script. The main steps that the Launcher has to perform, are:

- Loading the rule XML file received by the Scheduler
- Extracting the script included in the Rule (all the information included in the *Definition* section of the XML file)
- Initialising the JSEngine environment and preparing the script
- Calling the *Script Executor* for executing the script

Script Initialization – Before running the script, the Launcher checks and prepares the script code for the execution on JS Engine. In this phase, the JSEngine and the JavaScript script code are prepared according to the SpiderMonkey JS Engine guideline. All the functions related to AXMEDIS plug-ins and arguments of rule are created and initialised. The arguments initialisation allows defining global variables associated with the arguments used in the script by actualising them with values specified in XML rule description. In event of possible errors during the script initialization, a failure message is generated and sent to Executor Manager that routes it to the Scheduler.

6.9.3 Script Manager

The Script Manager is in charge to build the javascript code by assembling all embedded and referenced scripts provided by the AXCP Rule. In the debugging mode, the manager prepares breakpoints to be associated with traps. Each single script provides the list of own breakpoints and the manager uses to generate the full list of absolute references to the merged script. At the same time it associates new references with scripts and the relative line in each sigle source code. This allows at runtime a fast retrieval of the script and the line where the execution was brought and then use this information in the GUI to show the line of the source code. The script manager is also used to update breakpoints added or removed by the user for instance using the AXMEDIS Rule Editor.

6.9.4 JSENGINE (SpiderMonkey by Mozilla)

The AXMEDIS Rule Executor is equipped with a JavaScript Engine based on SpiderMonkey, the Mozilla's C implementation of JavaScript ([1]). The JS engine supports JS 1.0 through JS 1.6 JS 1.5, JS 1.4., JS 1.3 and greater conform to the ECMAScript-262 specification. At its simplest, the JS engine parses, compiles, and executes scripts containing JS statements and functions. The engine handles memory allocation for the JS data types and objects needed to execute scripts, and it cleans up--garbage collects--the data types and objects in memory that it no longer needs. Generally, the JS engine is built as a shared resource. For example, the engine is a DLL on Windows and Windows NT, and a shared library on Unix. The JS engine's API provides functions that fall into the following broad categories:

- Data Type Manipulation
- Run Time Control
- Class and Object Creation and Maintenance
- Function and Script Execution
- String Handling
- Error Handling
- Security Control
- Debugging Support

Conceptually, the JS engine is a shared resource on the system. By embedding engine API calls in the application, requests can be passed to the JS engine for processing. The engine, in turn, processes requests, and returns values or status information back to the application. The following picture illustrates this general relationship:

In truth, the actual relationship between the application and the JS engine is somewhat more complex than shown in Figure 1.1. For example, it assumes that you have already built the JS engine for your platform. It assumes that your application code includes `jsapi.h`, and it assumes that the first call your application makes to the engine initialises the JS run time.

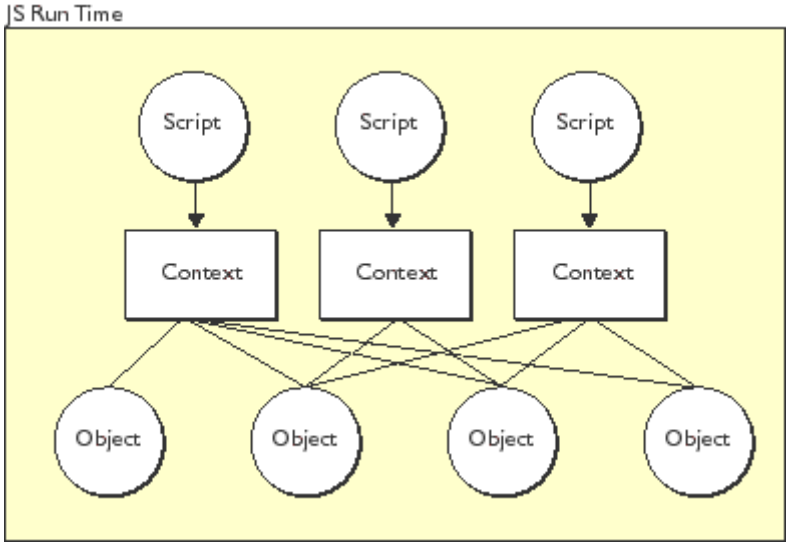
When the JS engine receives an initialisation request, it allocates memory for the JS run time. The picture reported below illustrates this process:

The run time is the space in which the variables, objects, and contexts used by the application are maintained. A context is the script execution state for a thread used by the JS engine. Each simultaneously existent script or thread must have its own context. A single JS run time may contain many contexts, objects, and variables.

Almost all JS engine calls require a context argument, so one of the first things the application must do after creating the run time is call `JS_NewContext` at least once to create a context. The actual number of contexts the application needs depends on the number of scripts expected to use at the same time in the

application. One context is needed for each simultaneously existing script in the application. On the other hand, if only one script at a time is compiled and executed by application, then you need only create a single context that you can then reuse for each script. After contexts creation, the built-in JS objects in the engine must be initialized by calling `JS_InitStandardClasses`. The built-in objects include the `Array`, `Boolean`, `Date`, `Math`, `Number`, and `String` objects used in most scripts.

Most applications will also use custom JS objects. These objects are specific to the needs of applications. They usually represent data structures and methods used to automate parts of your application. To create a custom object, you populate a JS class for the object, call `JS_InitClass` to set up the class in the run time, and then call `JS_NewObject` to create an instance of your custom object in the engine. Finally, if your object has properties, you may need to set the default values for them by calling `JS_SetProperty` for each property. Even though you pass a specific context to the JS engine when you create an object, an object then exists in the run time independent of the context. Any script can be associated with any context to access any object. The following figure illustrates the relationship of scripts to the run time, contexts, and objects.



As the previous figure also illustrates, scripts and contexts exist completely independent from one another even though they can access the same objects. Within a given run time, an application can always use any use any unassigned context to access any object. There may be times when you want to ensure that certain contexts and objects are reserved for exclusive use. In these cases, create separate run times for your application: one for shared contexts and objects, and one (or more, depending on your application's needs) for private contexts and objects.

NOTE: Only one thread at a time should be given access to a specific context.

6.9.5 Script Executor

The Script Executor receives the script code and arguments (Selections and parameters), then, it performs the necessary operations for:

- Invoking and initialising the JS Engine and variables.
- Sending the script to the JS Engine.
- Running and managing the communication with the JS Engine **according to the capabilities and functionalities provided by the JS Engine.**
- Routing errors coming from the JS Engine to the Rule Executor Manager.
- Sending Messages coming from the script in execution to the Rule Executor Manager.

The Script Executor will be developed to be used also in the AXCP Rule Editor. As depicted in the software architecture, the editor will be equipped with a Rule Executor. When used inside the AXCP Rule Editor it will be able to work also in two different modes: the rule debugging mode and rule check mode. This will be useful during the definition of a rule since the user will be able to test the rule and to solve possible errors

Script Executor: Debugging Mode
The debugging mode was realised by using the debug function provided by JSDebug API of SpiderMonkey and to be controlled by the AXCP Rule Editor. The Spidermokey APIs permit to :

- put traps in the code corresponding to breakpoints (interrupting the execution)
- defining callbacks for:
    - intercepting the execution of javascript bytecodes
    - intercepting the call of scripted functions
    - intercepting the allocation of new objects
    - intercepting the deallocation of objects and the internal Garbace Collector
- watch variables
- manage the stack of functions
- realise the interface for debug functions and controls for the AXCP Rule Editor.

Script Executor: Check Mode
This modality is mainly used by AXCP Rule Editor when it is necessary to check the feasibility of a rule. In the testing mode, the rule is executed in order to:

- verify the correctness of the rule before to send it to the AXCP Rule Engine
- estimate some parameters related to the complexity of the rule. Such parameters will be identified and defined during the project life. They will be used to define a complete profile of the rule in terms of required computational resources.

6.9.6 Executor Profile and XML formalisation

The executor profile is the set of metadata that allows to describe the executor in terms of:

1. Computational capabilities
2. Functionalities that the executor provides such as:
    - AXMEDIS Plug-In installed.
    - External tools Plug-In installed

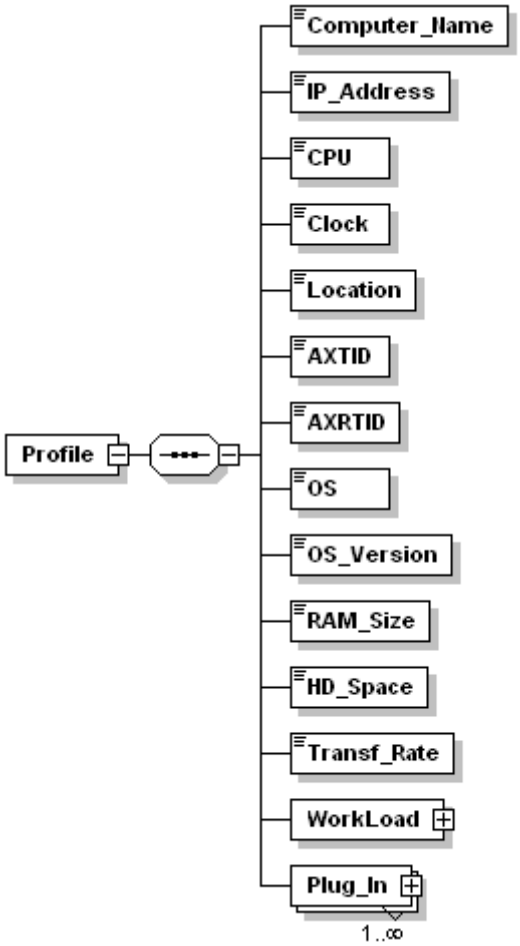
The following table describes each metadata of profile:

Attributes	Description
Computer Name	Name of the computer hosting the rule executor
URL	IP address
AXTID	The ID of a specific instance related to the AXTID
AXRTID	The AXMEDIS Registered Tool Id associated with the Rule Executor application
SO	The Operating System
Version	The SO version
CPU	The type of CPU
Clock	The clock of CPU
RAM size	The amount of memory
HD-space	The amount of disk space available
Location	Where the computer is located
Workload	The percentage of availability during the time period
Transfer rate from AXDB	The network capability for transferring a file from the AXMEDIS Database to the rule executor machine.
Plug-In	It provides the name and the version. Since the executor can host many plug-in, this is field represents a list of Plug-In

The definition of profile for a rule executor peer was formalised by means the XML Schema as reported in this section. The profile describes the capability of the machine where the rule executor peer is running in terms of hardware, software and axmedis plugin configuration.

element **Profile**

diagram



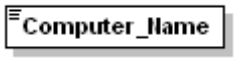
children Computer_Name IP_Address CPU Clock Location AXTID AXRTID OS OS_Version RAM_Size HD_Space Transf_Rate WorkLoad Plug_In

source

```
<xs:element name="Profile">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Computer_Name" type="xs:string"/>
   <xs:element name="IP_Address" type="xs:anyURI"/>
   <xs:element name="CPU" type="xs:string"/>
   <xs:element name="Clock" type="xs:float"/>
   <xs:element name="Location" type="xs:string"/>
   <xs:element name="AXTID" type="xs:string"/>
   <xs:element name="AXRTID" type="xs:string"/>
   <xs:element name="OS" type="xs:string"/>
   <xs:element name="OS_Version" type="xs:string"/>
   <xs:element name="RAM_Size" type="xs:string"/>
   <xs:element name="HD_Space" type="xs:string"/>
   <xs:element name="Transf_Rate" type="xs:unsignedInt"/>
   <xs:element name="WorkLoad">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Percentage" type="xs:float"/>
      <xs:element name="Start_Time" type="xs:time"/>
      <xs:element name="End_Time" type="xs:time"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="Plug_In" maxOccurs="unbounded">
    <xs:complexType>
```

```
            <xs:sequence>
              <xs:element name="Name" type="xs:time"/>
              <xs:element name="Version" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

element **Profile/Computer_Name**

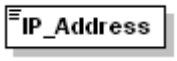
diagram 

type **xs:string**

source `<xs:element name="Computer_Name" type="xs:string"/>`

description Name of the computer hosting the rule executor

element **Profile/IP_Address**

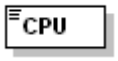
diagram 

type **xs:anyURI**

source `<xs:element name="IP_Address" type="xs:anyURI"/>`

description IP address

element **Profile/CPU**

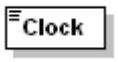
diagram 

type **xs:string**

source `<xs:element name="CPU" type="xs:string"/>`

description The type/family of CPU

element **Profile/Clock**

diagram 

type **xs:float**

source `<xs:element name="Clock" type="xs:float"/>`

description The clock of CPU

element **Profile/Location**

diagram 

type **xs:string**

source `<xs:element name="Location" type="xs:string"/>`

description Where the computer is located

element **Profile/AXTID**

diagram 

type **xs:string**

source <xs:element name="AXTID" type="xs:string"/>

description The ID of a specific instance related to the AXTID

element **Profile/AXRTID**

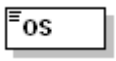
diagram 

type **xs:string**

source <xs:element name="AXRTID" type="xs:string"/>

description The AXMEDIS Registered Tool Id associated with the Rule Executor application

element **Profile/OS**

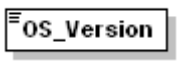
diagram 

type **xs:string**

source <xs:element name="OS" type="xs:string"/>

description The Operating System

element **Profile/OS_Version**

diagram 

type **xs:string**

source <xs:element name="OS_Version" type="xs:string"/>

description The OS version

element **Profile/RAM_Size**

diagram 

type **xs:string**

source <xs:element name="RAM_Size" type="xs:string"/>

description The amount of primary memory (RAM)

element **Profile/HD_Space**

diagram 

type **xs:string**

source <xs:element name="HD_Space" type="xs:string"/>

description The amount of disk space available

element **Profile/Transf_Rate**

diagram

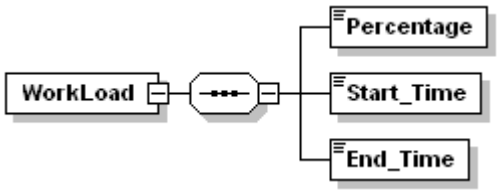
type **xs:unsignedInt**

source
```
<xs:element name="Transf_Rate" type="xs:unsignedInt"/>
```

description The network capability for transferring a file from the AXMEDIS Database to the rule executor machine.

element **Profile/WorkLoad**

diagram



children **Percentage Start_Time End_Time**

source
```
<xs:element name="WorkLoad">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Percentage" type="xs:float"/>
      <xs:element name="Start_Time" type="xs:time"/>
      <xs:element name="End_Time" type="xs:time"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

description The percentage of availability during the time period

element **Profile/WorkLoad/Percentage**

diagram

type **xs:float**

source
```
<xs:element name="Percentage" type="xs:float"/>
```

description Availability to work expressed in percentage

element **Profile/WorkLoad/Start_Time**

diagram

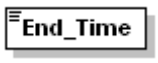
type **xs:time**

source
```
<xs:element name="Start_Time" type="xs:time"/>
```

description Start time of declared workload percentage

element **Profile/WorkLoad/End_Time**

diagram



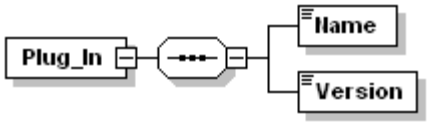
type **xs:time**

source
```
<xs:element name="End_Time" type="xs:time"/>
```

description End time of declared workload percentage

element **Profile/Plug_In**

diagram



children **Name** **Version**

source
```
<xs:element name="Plug_In" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:time"/>
      <xs:element name="Version" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

description Axmedis Plugin description mounted in the system

element **Profile/Plug_In/Name**

diagram

type **xs:time**

source `<xs:element name="Name" type="xs:time"/>`

description Name of plugin

element **Profile/Plug_In/Version**

diagram

type **xs:string**

source `<xs:element name="Version" type="xs:string"/>`

description Version of plugin

6.10 Configuration Parameters

In this section the set of parameters regarding the configuration of the editor are listed. Such parameters are grouped into modules as reported below:

6.10.1 AXMEDIS Rule Executor

Config parameter	Possible values
XML_RULE_PATH	it is the directory where the rule will be saved
XML_XSD_PATH	It is the directory where xml schema (XSD files) are stored
LOGS_PATH	It is the directory where logs are stored

6.10.2 AXMEDIS_GRID_SUPPORT_SETTINGS

Config parameter	Possible values
SERV_PORT	It is the value of the port used by the Grid Interface
SERV_PORT2	It is the value of the port used by the Grid Interface

SERV_PORT3	It is the value of the port used by the Grid Interface
SERV_PORT4	It is the value of the port used by the Grid Interface
SERV_PORT5	It is the value of the port used by the Grid Interface
DEFAULT_NET	It is the list of IP address

6.10.3 AXMEDIS Plugin Manager

Config parameter	Possible values
PLUGINS_PATH	It is the directory where the DLL of plug-ins with their profiles (workflow, adaptation, descriptor and fingerprint estimators) are stored.

6.10.4 AXMEDIS Database

Config parameter	Possible values
user	The user name for logging into Database
passwd	The passoword for logging into Database
LoaderWSEndPoint	It is the URL for…
HTTPPath	
UploadPath	
SaverWSEndPoint	

6.10.5 AXMEDIS Selection

Config parameter	Possible values
MAIN_QUERY_SUPPORT_WSDL	It is the URL of the WSDL for using the Main Query Support
SELECTION_ARCHIVE_WSDL	It is the URL of the WSDL for using the Selection Archive

Example of Configuration file

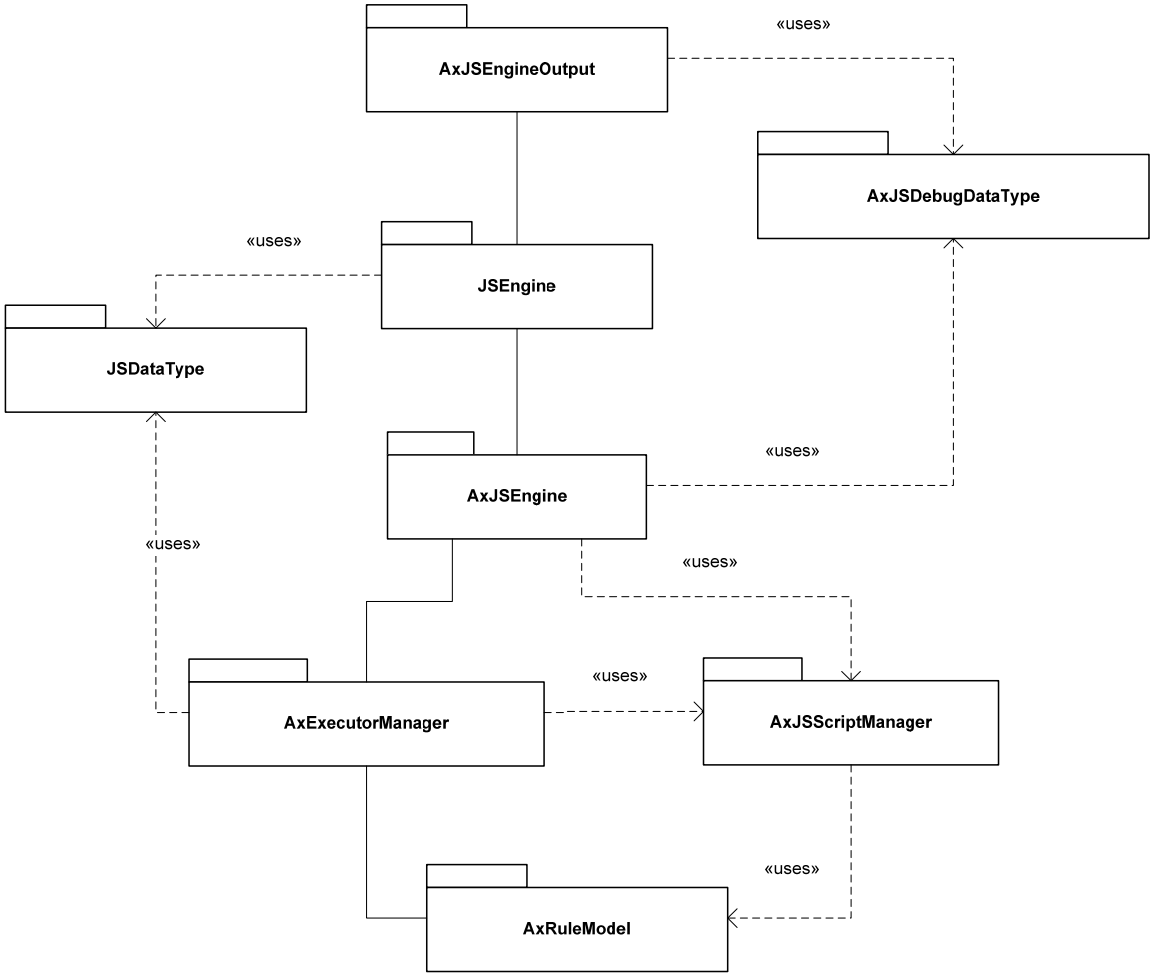
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Configuration xmlns="http://www.axmedis.org/configuration">

  <Module category="" id="AXMEDIS_RULE_EXECUTOR">
    <Parameter name="XML_XSD_PATH" type="string">./</Parameter>
    <Parameter name="XML_RULE_PATH" type="string">./Rules</Parameter>
    <Parameter name="LOGS_PATH" type="string">./Logs</Parameter>
  </Module>

  <Module category="" id="AXMEDIS_PLUGIN_MANAGER">
    <Parameter name="PLUGINS_PATH" type="string">./Plugin</Parameter>
  </Module>

  <Module category="" id="AXMEDIS_GRID_SUPPORT_SETTINGS">
    <Parameter name="SERV_PORT" type="string">3000</Parameter>
    <Parameter name="SERV_PORT2" type="string">3001</Parameter>
    <Parameter name="SERV_PORT3" type="string">3002</Parameter>
    <Parameter name="SERV_PORT5" type="string">3014</Parameter>
    <Parameter name="SERV_PORT4" type="string">3006</Parameter>
    <Parameter name="DEFAULT_NET" type="string">192.168.0.255</Parameter>
  </Module>

  <Module category="" id="DATABASE" visible="true">
    <Parameter name="user" type="string">test</Parameter>
    <Parameter name="passwd" type="string">test</Parameter>
    <Parameter name="LoaderWSEndPoint" type="string">http://bellini-
mobile:8080/LoadSaveWS/load</Parameter>
```

```
    <Parameter name="UploadPath"
type="string">C:\wedelmusic\localdistributor\htdocs\axdb-share</Parameter>
    <Parameter name="SaverWSEndPoint" type="string">http://bellini-
mobile:8080/LoadSaveWS/save</Parameter>
    <Parameter name="FTPPath" type="string">ftp://bellini-mobile/</Parameter>
  </Module>

  <Module category="" id="AXMEDIS_SELECTION">
    <Parameter name="MAIN_QUERY_SUPPORT_WSDL" type="string">http://bellini-
mobile:8080/MainQuerySupportWS/mqs?WSDL</Parameter>
    <Parameter name="SELECTION_ARCHIVE_WSDL" type="string">http://bellini-
mobile:8080/UserSelectionArchiveWS/sa?WSDL</Parameter>
  </Module>


</Configuration>
```

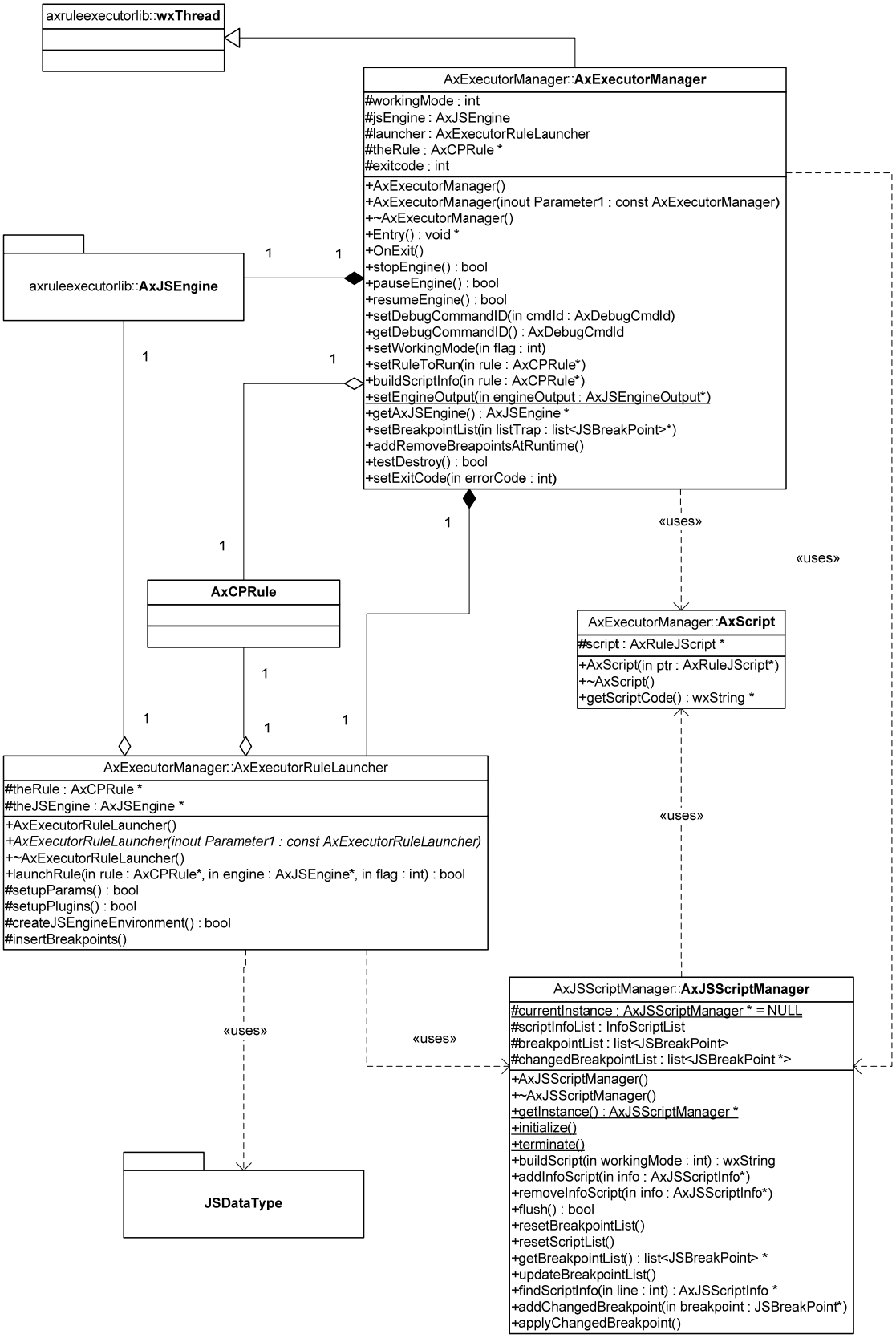
6.10.6 Rule Executor Class Diagrams

The following class diagram shows main components of the Rule Executor. The diagram provides a view of executor core system:

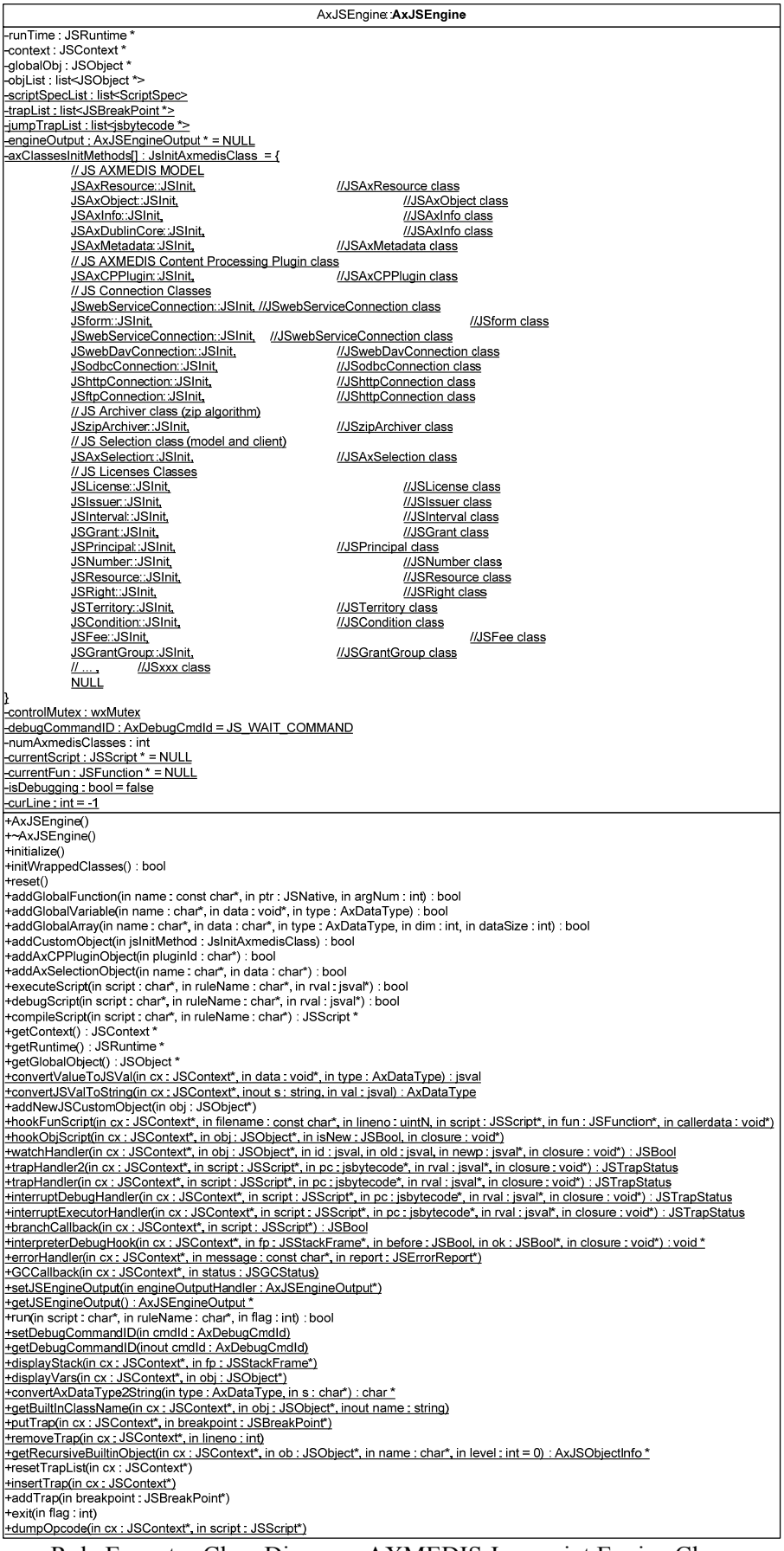
- **JSEngine** – It is the SpiderMonkey JavaScript Engine. It is the set of APIs provided by the Library
- **AxJSEngine** – It is the component that models the Script Executor
- **AxExecutorManager** – It is the component that models the *Executor Manager* and *Launcher*
- **AxJSScriptManager** – It is the component that models the *Script Manager*
- **AxJSEngineOutput** – It is the component that models the *Engine Output*
- **AxJSDataType** – It is the componet that collects all JS Data Type derived from the AXMEDIS Framework and extends the built in classes provided by the JSEngine



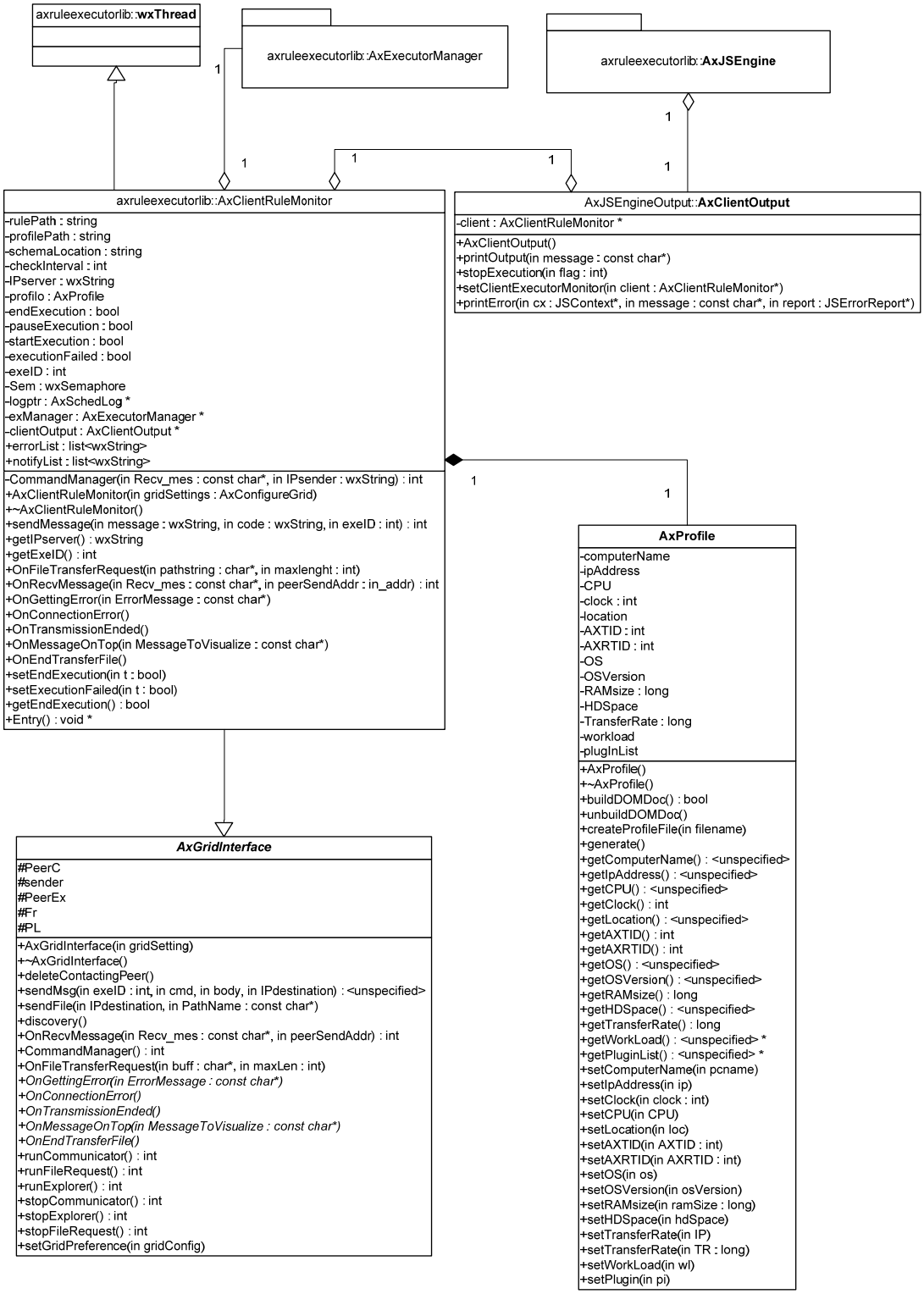
Rule Executor Class Diagram: Main Componenents



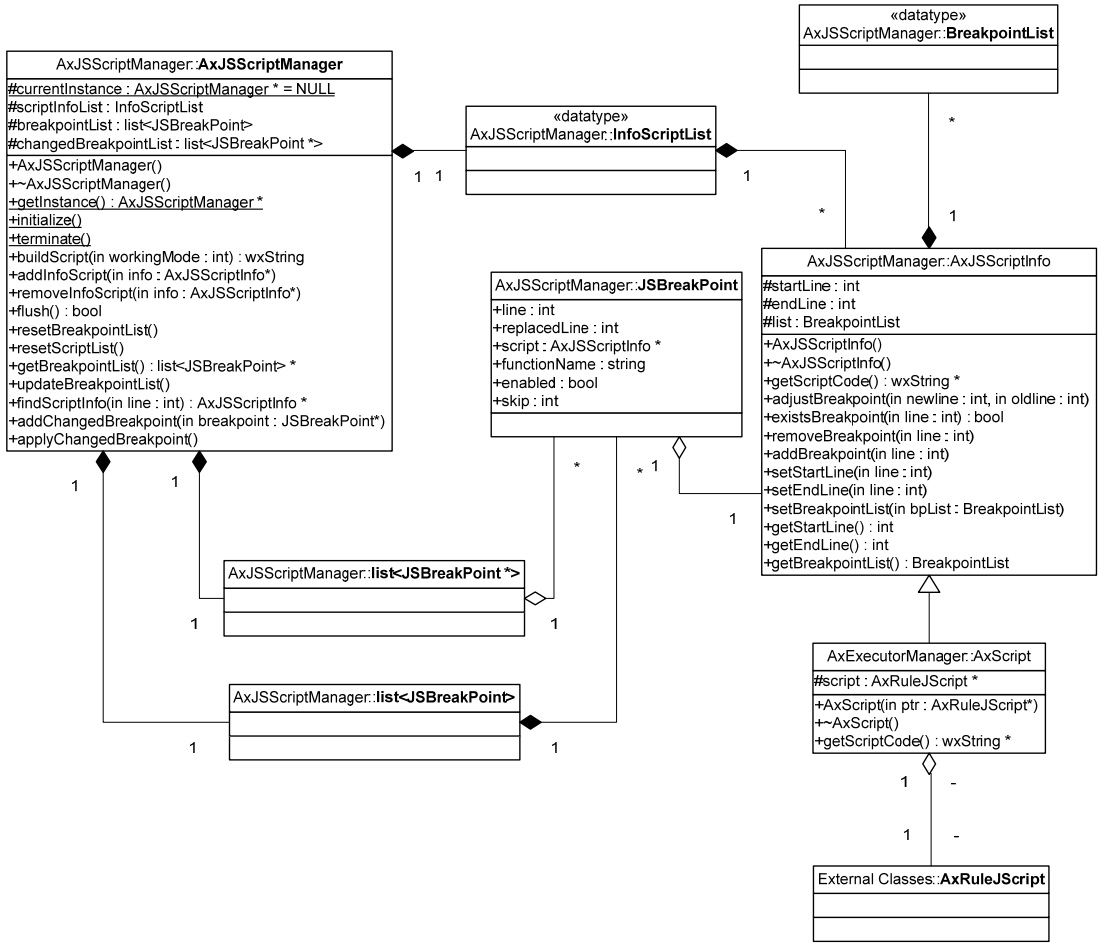
Rule Executor Class Diagram: Executor Manager & Launcher



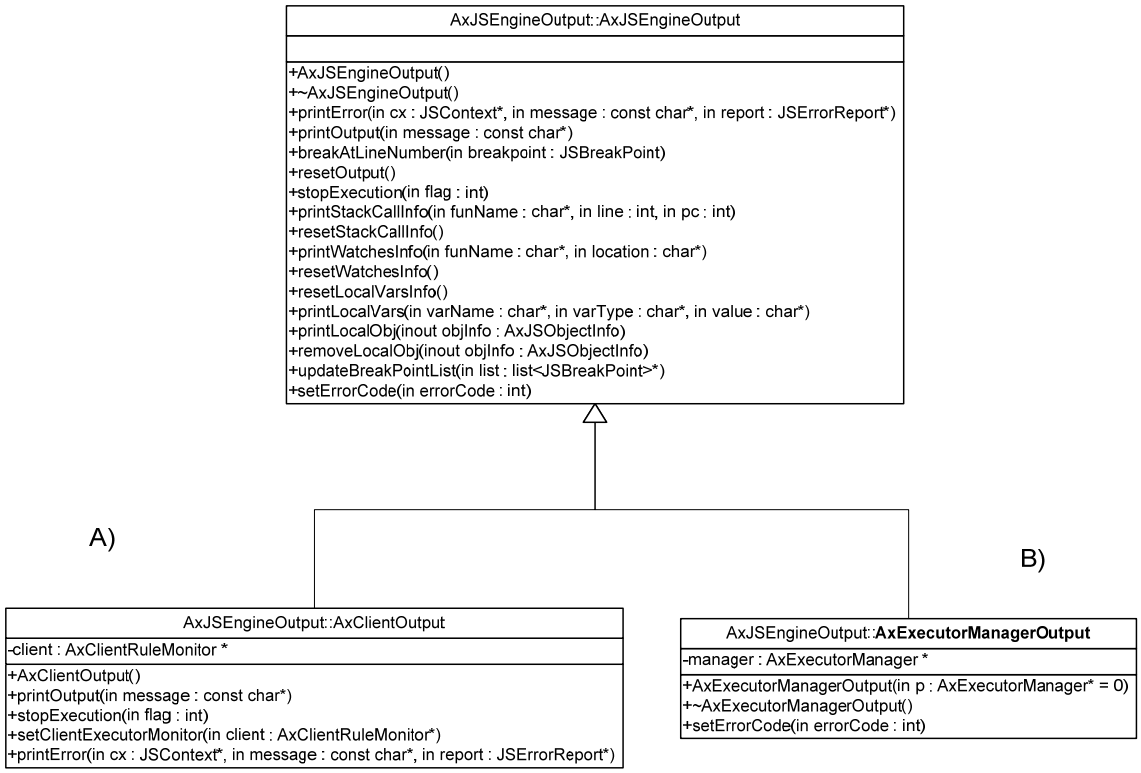
Rule Executor Class Diagram: AXMEDIS Javascript Engine Class



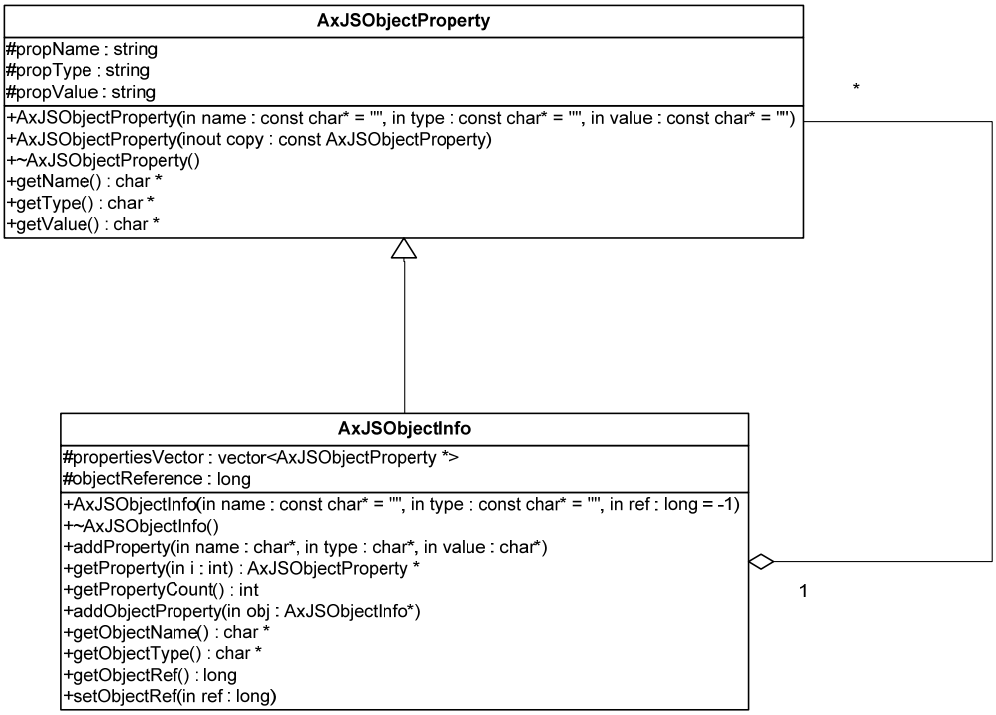
AXCP Grid Node



AxJSScriptManager Class Diagrams



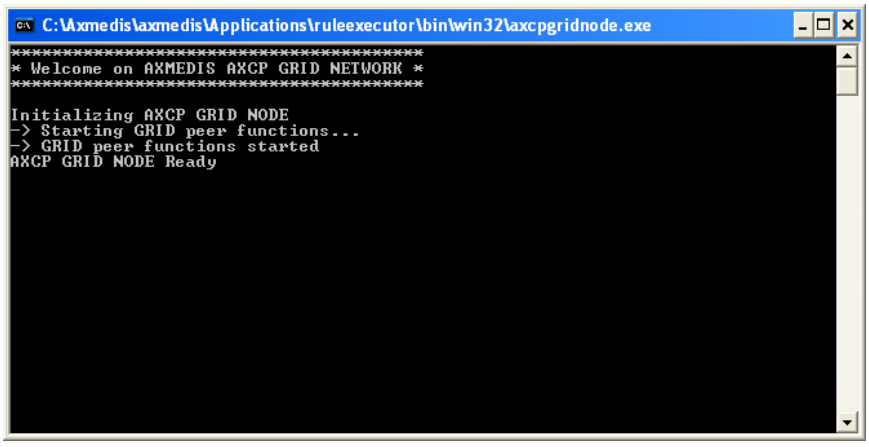
AxJSEngine Output System: A) output for GRID; B) output for Stand Alone



AxJSDebugDataType Class Diagram

6.11 User interface description

The User Interface of the rule executor as GRID Node is shown in the following picture. It is a console application and the output of the engine for direct messages.



6.12 Technical and Installation information

References to other major components needed	The AXCP Grid Node works in conjuction with the AXCP Rule Scheduler.
Problems not solved	The current version is not equipped with a system to limit the request of CPU. So it could happen that the run of a rule requires 100% of CPU. This will be solved in the next version.
Configuration and execution context	The Application provides a configuration file.

6.13 Draft User Manual

The AXCP Rule Executor is provided as two executable files:

- Stand alone AXCP Executor as console application for running AXCP Rule
- AXCP Grid Node Executor as console application for the GRID Environmet

Before the first launch of the application, the user should setup the configuration file in order to provide the right value to the constants and parameters used by the application. See the section of the Confiuration Parameters

6.14 Examples of usage

Stand alone AXCP Executor

To run the Executor with an AXCP Rule, the user has to open the CMD dialog of Windows and type the line command using the option 0 as following:

axruleexecutor.exe <rule path> -0

AXCP Grid Node Executor

To launch the AXCP Grid Node, the user has to double click on axcpgridnone.exe executable file

6.15 Integration and compilation issues

The Visual Studio solution provides to project for executable files.

6.16 Errors reported and that may occur

Not available at this step.

Error code	Description and rationales

7 AXMEDIS Content Format Engine

Module Profile		
AXMEDIS Content Format Engine		
Responsible Name	Paolo Vaccari	
Responsible Partner	DSI	
Status (proposed/approved)		
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Multi Thread	
Language of Development	C++	
Platforms supported	MS Windows	
Reference to the AXFW location of the source code demonstrator		
Reference to the AXFW location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)	yes	
Usage of the AXMEDIS Error Manager (yes/no)		
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Resource descriptor		Resource descriptor
Template descriptor		Template descriptor

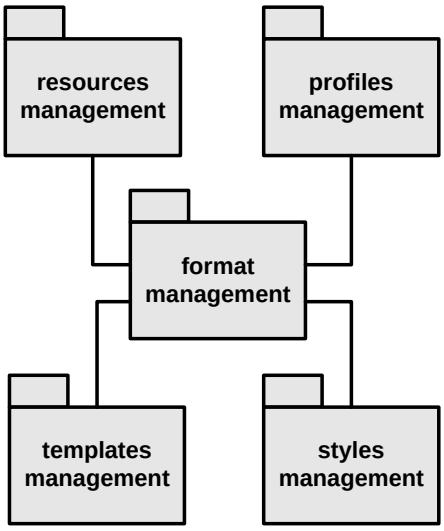
Style descriptor		Style descriptor
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces	Xerces-C++ v.2.7	LGPL
Xalan	Xalan-C++ v.1.10	LGPL
GALib	GALib v.2.4.6	BSD-like

7.1 General Description of the Module

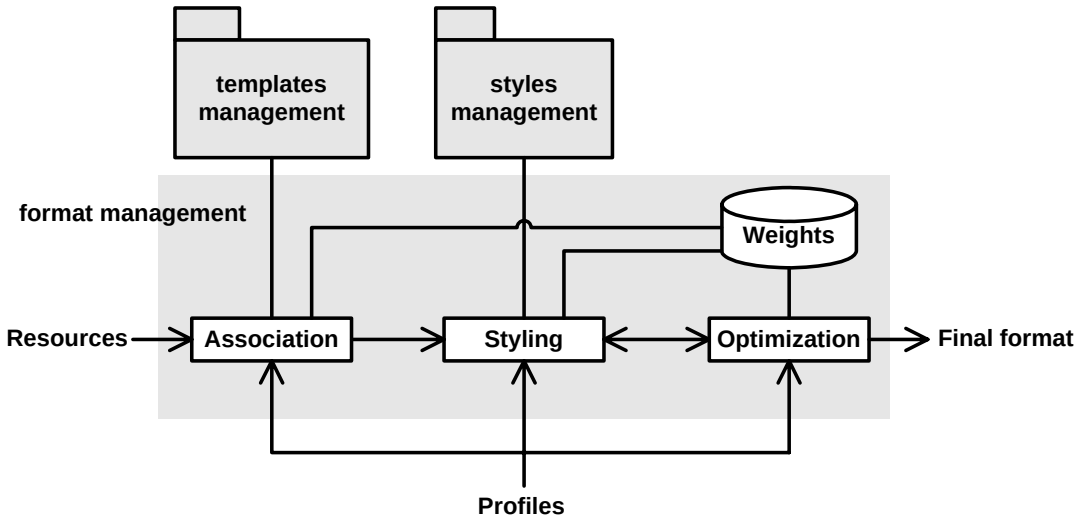
The Content Format Engine provides formatting functionalites that are used by the AXCP GRID and by format tools (*Template Editor and Selector*, *Style Editor and Selector*, *Style Optimizer*) integrated within the AXMEDIS SMIL Editor.

The goal of the Content Format Engine is integrating digital resources, contained within an AXMEDIS Object, in a multimedia presentation (based on the SMIL language) suitable for the final user. The Engine is based on templates and style-sheets that define respectively the basic structure and the look-and-feel of the presentation. To get best results in terms of adaptation to customer's needs, the Content Format Engine performs its choices according to profiles that describe personal preferences, user's device capabilities and delivery context.

The Content Format Engine Module is composed of five main subsystems:



Resources and profiles (user, device and context profiles) represent the input to the Format Engine; templates and styles are its output; the format management implements the filtering and optimization logic.
The format management subsystem is logically divided into three blocks:



- the **association** block receives as input the digital resources, profiles, and some additional information that provides basic knowledge about the type of the presentation, its output format and the target platform. It performs a mapping of resources, profiles and other compositional properties, and chooses most suitable templates, according to a set of weights. The output of this block is the indication of a template, which describes the basic structure of the presentation;
- the **styling and optimization** blocks select a style-sheet for the given template and adjusts their parameters to cope with the profiles, managing the adaptation and the transcoding of the involved media. They perform a mapping of resources, profiles and other compositional properties, and choose most suitable style-sheets and optimization values, according to a set of weights. The output of this block is the final format description.
  The optimization process is a very critical and complex phase, that may involve many aspects. Media encoding is part of the optimization: to fit the optimized layout, media may necessitate transcoding, resampling or moreover, media have to be transformed (scaled, rotated, etc.) to fit the adapted layout.
  The choice of the optimization algorithm is very important: the problem of finding the best combination of a potentially large number of layout parameters is NP-complete, therefore the computational time to determine an exact solution is not reasonable. A more practical approach is to

search “good” solutions, with methods such as Genetic Algorithms that can be distributed on the AXMEDIS GRID.

The system is supposed to work in two modalities:

- an interactive modality, which allows the author to choose or create templates and style-sheets and control the results of the adaptation. This modality also relies on the AXMEDIS SMIL Editor and Player;
- an automatic modality, that manages the whole process following a set of rules; these rules use the JavaScript modules that wrap the Format Engine.

In either the interactive or the automatic modality, the *format_manager* is used to set profiles and resources, that are the input of the Format Engine. The input is used by the Association block to associate the document with an existing template. The Association block can also interact with the user: the system either requires a choice between different options (if several templates match the input) or allows the creation of a new template.
To correctly associate the input set with a suitable template, the multimedia resources should be ordered or logically grouped, according to their semantic or spatial relationships, using the hierarchical organization of the AXMEDIS Object. Such relationships may also reflect the logic within the querying process.
After that, the style-sheet selection is performed by the Styling/Optimization block: in the automatic modality one of the style-sheets created for the given template is selected following profiles indications; otherwise, a filtered list of style-sheets is proposed to the author, who can choose one of them or create a new one. The final stage is the optimization: the parameters of the style-sheet are optimized following the profiles specification, and all media are transcoded and transformed to fit the final layout.

7.1.1 Resources management

Resources that have to be formatted may come from various sources:

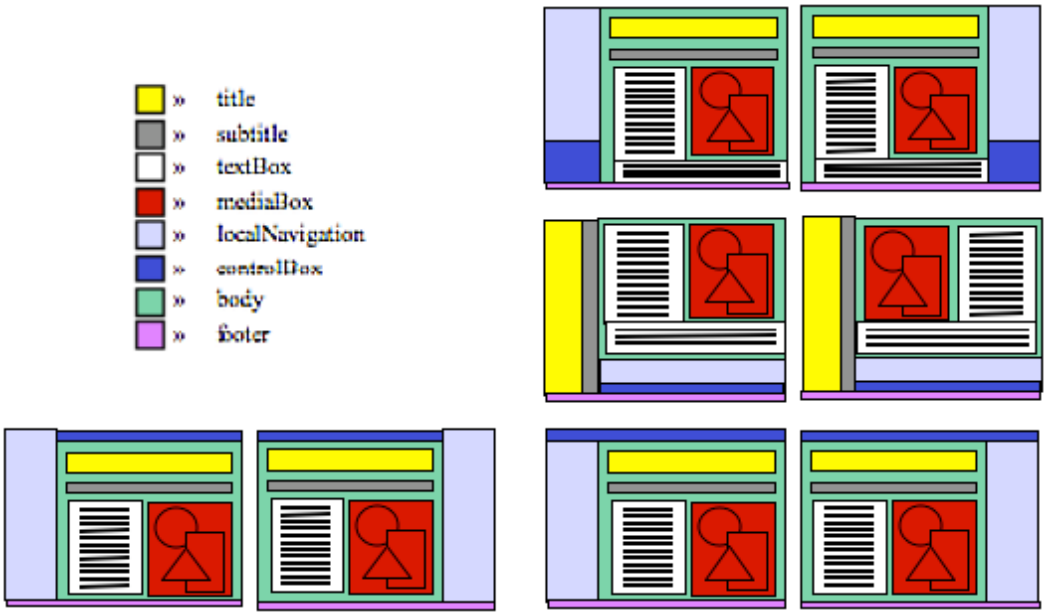
- resources or references included into an AXMEDIS object;
- list of resources accessible in the local file system or through an URI;
- references resulting from a query;
- references returned from an external library.

Anyway, resources may be represented by an XML descriptor. Descriptors are not stored, but only created in memory during the formatting process.
The ***resource_descriptor*** specifies:

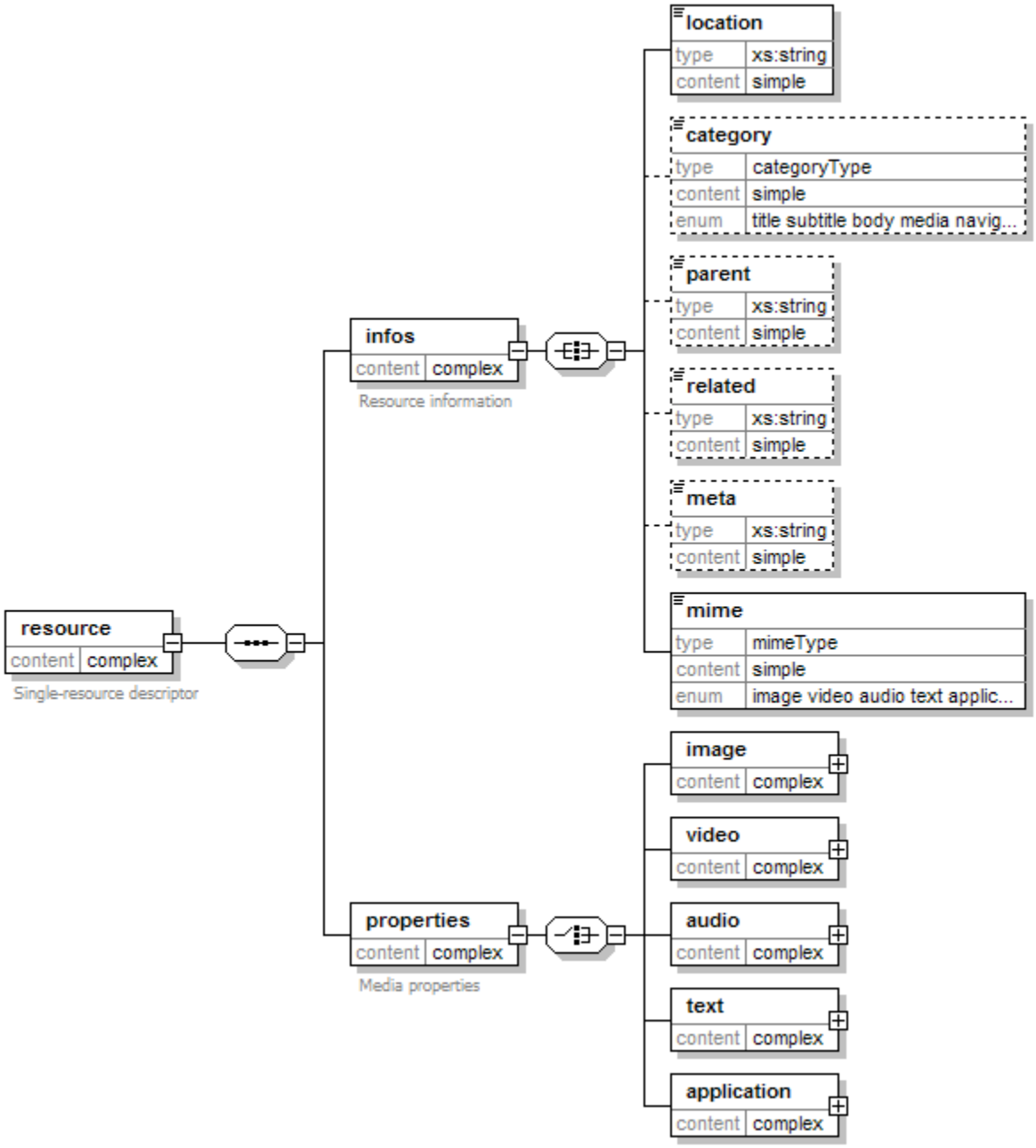
- id: identifier;
- location: the resource address (URI, file system location, AXMEDIS Object ID (AXOID), etc.). Regarding the AXObjects, we consider that each single resource is encapsulated into a single AXObject and thus identified through its AXOID; by the way, the single-resource AXObject may be encapsulated into a composite AXObject;
- type: the MIME top-level type of the resource (i.e.: image, audio, text, video, or application);
- subtype: the MIME subtype (e.g.: bmp, jpg, wav, mp3, rtf, txt, etc.);
- meta (optional): additional information related to the resource;
- parent (optional): the reference of the parent resource. Typically, it will be used for navigate into a hierarchy of encapsulated AXObjects (presently not used);
- related (optional): the reference of another resource which has strong logical relationship with the current resource (presently not used);
- properties: depending on the type, specifies available attributes (size, resolution, codec, encoding, etc.);

- category (optional): the way we intend to use the resource. For instance: a text may be used as document title, as page body, as figure caption, as button label; an image may be used as page header, as background, as figure; and so on.

  For defining resource categories, we consider a typical web-based content page that holds a set of content comprising: title, sub-title (if needed), body text (resizable and scrollable in respect to display area), some multimedia content (with related controls when needed), a quick navigation bar (if needed to allow content browsing) and potentially comprising or associated with a control bar (to allow overall content management: print, save…) and a footer usually holding copyright information. The usual structures are represented in the following picture.



The XML Schema for the resource descriptor is depicted below.



7.1.2 Templates management

Each template has an associated descriptor, that resumes its principal characteristics. Descriptors are stored locally and refer to templates that may be stored elsewhere. A ***template_descriptor*** contains some information that comes from a template analysis, and other inputs from the template author.
With a template analysis we can get:

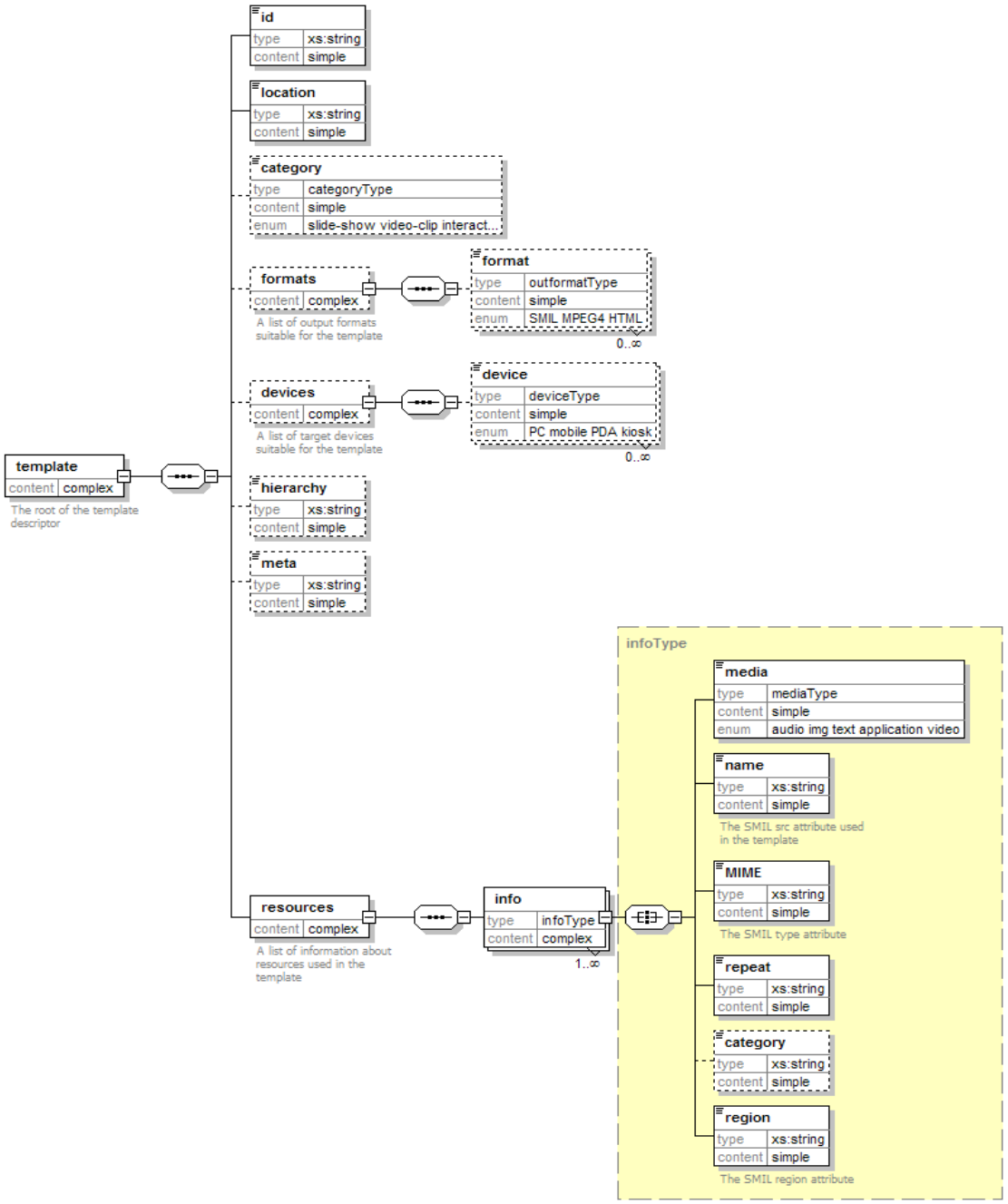
- type of the resources needed by the template, and their number;
- hierarchical (tree-shaped) organization of the resources within the template;
- category indications provided by the author for each resource (optional).

The indications that the template author has to supply are:

- ID;
- general category for the template, chosen from a predefined list (e.g.: slide show, video clip, interactive music, interactive video, hypertext, audio/video karaoke, electronic book, kiosk, training tutorial, etc.);

- output format (e.g.: SMIL, MPEG4, HTML, etc.). Some templates could include elements and/or structures that may not be rendered in all formats; for instance, SMIL and MPEG4 are not fully interchangeable, and HTML is really poor if compared to SMIL.
- target devices (e.g.: PC, mobile, kiosk, PDA, etc.);
- metadata, with a general template description.

The XML Schema for the template descriptor is depicted below.



7.1.3 Templates filtering

The ***template_filter*** performs various types of ranking for the templates in the *templates_list*. Each element of these filtered lists is a reference to the an element of the complete *templates_list*; moreover, each filtered list may be used as source for a new filtering operation.

Filtering operations are based on criteria such as:

- general category, target devices and output format for the template, compared with author inputs;
- correspondence between type and number of resources needed by the templates and the *resources_list*;
- correspondence between hierarchical organizations of template and resources;
- correspondence between category attributes for resources and template.

The relevance of the criteria is determined by weights, selected from a list of possibilities (managed by a *format_criteria* and stored in a *Weights* table): the author may choose the preferred set of weights or define a new one.
Once we have evaluated the criteria we have to normalize their values (C1, ..., Cn) between 0 and 1; then we can get the normalized score of a template:

$$score(T) = (W1\ C1 + W2\ C2 + ... + Wn\ Cn) / (W1 + W2 + ... + Wn)$$

7.1.4 Style-sheets management

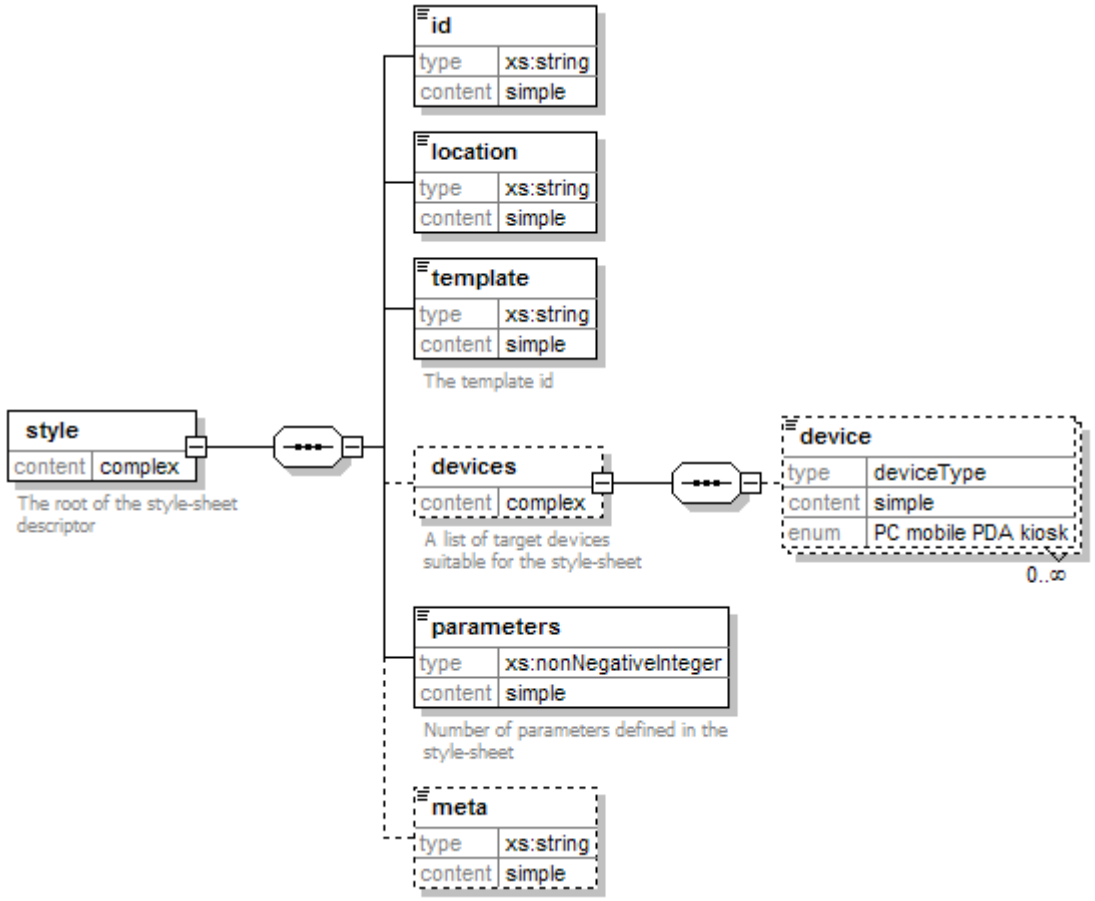
Each style-sheet has an associated descriptor, that resumes its principal characteristics. A ***style_descriptor*** contains some information that comes from a style-sheet analysis, and other inputs from the style-sheet author.
With a style-sheet analysis we can get:

- parameters specified for the optimization.

The indications that the template author has to supply are:

- ID;
- ID of the related template;
- target devices (optional), that may be a subset of devices specified in the template;
- metadata (optional), with a general style-sheet description.

The XML Schema for the style-sheet descriptor is depicted below.



7.1.5 Style-sheets filtering

The ***style_filter*** performs various types of ranking for the style-sheets in the *styles_list*. Each element of these filtered lists is a reference to the an element of the complete *styles_list*; moreover, each filtered list may be used as source for a new filtering operation.
Furthermore, this class uses the *optimizer* to get optimized values for parameters defined in the style-sheets.
Filtering operations are based on:

- related template;
- target devices;
- need for optimization (presence of parameters).

The relevance of the criteria is determined by weights, selected from a list of possibilities (managed by a *format_criteria* and stored in a *Weights* table): the author may choose the preferred set of weights or define a new one.
Once we have evaluated the criteria we have to normalize their values (C1, ..., Cn) between 0 and 1; then we can get the normalized score of a template:

$$score(T) = (W1\ C1 + W2\ C2 + ... + Wn\ Cn) / (W1 + W2 + ... + Wn)$$

7.1.6 Style-sheet optimization

The ***optimizer*** class manages the document optimization. This task is done assigning values to the parameters specified in the style-sheets: a choice of values that follow the profiles, allows a customization of the layout and of the general look-and-feel of the document. Such an optimization is an hard problem that may involve many parameters; the search space is large and it is unlikely to find an optimal solution, thus metaheuristic algorithms can get good results.

Genetic Algorithms (GA) are adaptive metaheuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. Possible solutions are considered as a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and recombination (crossover). Initially many individual solutions are randomly generated to form an initial population. At each successive step, a part of the existing population is selected to breed a new generation: solutions are evaluated by a fitness function and stochastic methods, that select best individuals. This generational process is repeated until a termination condition (minimum criteria, number of generations, computational time, etc.) has been reached.
Typically, parameters defined in the style-sheets will be position and dimensions of the regions composing the layout. The search of best layout can be modeled as a 2D bin-packing problem of rectangular shapes on a rectangular canvas and solved using the GA approach.
Other parameters could be: screen dimensions, text properties (font size, alignment, etc.), timing, colors, etc. The fitness function of GA should consider several aspects: regions overlapping, alignment and maximization, minimization of the unused space, text readability, etc. The relevance of each aspect is evaluated according to a set of weights (managed by a *format_criteria* and stored in a *Weights* table).

7.1.7 Criteria

The **format_criteria** class manages weights used for template filtering, style-sheet filtering and optimization. Each set of weighs is stored together with an unique id, a description (metadata), and its type (template, style or optimizer).

7.1.8 Profiling

The Content Format Engine needs information about the destination of the document that has to be formatted, to perform best choices during selection and optimization. Such information should be grouped in three profiles:

1. user profile;
2. device profile;
3. context (network) profile.

Profiles may be static or dynamic and may depend on the category of the client device (mainly PC or mobile):

- information about users is collected statically and does not depend on the device category;
- information about device is collected both statically and dynamically and depends on the device category;
- information about context is collected dynamically and does not depend on the device category.

The AXMEDIS Device profile and Context profile are subsets of MPEG-21 DIA while User profile contains some elements of MPEG-21 DIA and some additional AXMEDIS elements in it.

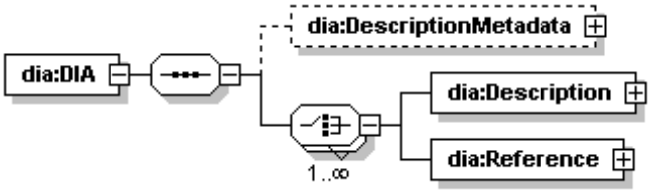
MPEG-21 Digital Item Adaptation specifies the syntax and semantics of tools that may be used to assist the adaptation of Digital Items; these tools are clustered into eight major categories. We are specially interested in the Usage Environment Description (UED) Tools, which include User characteristics, terminal capabilities, network characteristics and natural environment characteristics. These tools provide descriptive information about the various properties of the usage environment, which originate from Users, to accommodate, for example, the adaptation of Digital Items for transmission, storage and consumption.

The Schema used for descriptions in MPEG-21 DIA defines:

1. the root elements;
2. the base types that form the hierarchy of the description.

The functionality of a root element is to describe the initial wrapper or root element of descriptions. Two root elements are defined:

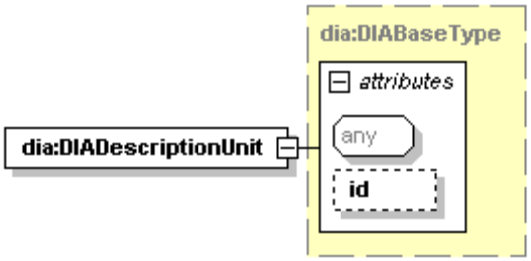
1. a *DIA* root element, which is used for complete descriptions:



   The *DescriptionMetadata* element can be used to contain the metadata for the descriptions contained within the DIA description.

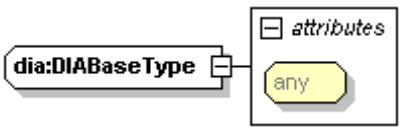
   Description and *Reference* are elements (or references to elements) whose type is derived from *DIADescriptionType*.

2. a *DIADescriptionUnit* root element, which is used to represent partial information from a complete description:

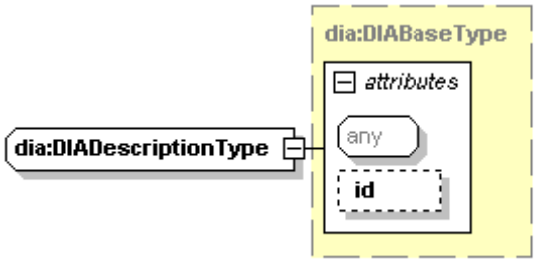


Two base types are defined:

1. a *DIABaseType*, that provides the base abstract type of the type hierarchy:



2. a *DIADescriptionType*, that extends *DIABaseType* and provides the base abstract type for a subset of types that may follow the *DIA* root element:



7.1.9 User profile

User characteristics are described in terms of:

- general information (e.g.: name, contact info, etc.);
- usage preferences and history;
- presentation preferences (audio, e.g.: equalizer settings, frequency, volume, etc.; display, e.g.: color temperature settings, contrast, brightness, etc.);
- accessibility (auditory and visual impairments);
- location.

Using DIA types and subtypes, the information mainly required by the Content Format Engine is:

- *UserInfo*: specifies general information about users, such as name and contact information;
- *AudioPresentationPreferences*: specifies the preferences of a user regarding the presentation or rendering or audio resources (such as the preferred volume, frequency equalizer settings, and audible frequency);
- *DisplayPresentationPreferences*: specifies the preferences of a user regarding the presentation or rendering of images and videos;
- *ColorPreference*: describes preferences related to color attributes;
- *GraphicsPresentationPreferences*: specifies presentation preferences related to graphics media;
- *ConversionPreference*: enables users to specify preferences to guide the conversion of resources when a terminal or network cannot support the consumption or transport of a particular modality or format;
- *PresentationPriorityPreference*: lets the user have choices on the presentation qualities of different resources at the output of the content adaptation process.

An example of user profile is the following:

```
<DIA>
   <Description xsi:type="UsageEnvironmentType">
      <UsageEnvironmentProperty xsi:type="UsersType">
         <User>
            <UserCharacteristic xsi:type="UserInfoType">
               <UserInfo xsi:type="mpeg7:PersonType">
                  <mpeg7:Name>
                     <mpeg7:GivenName>John</mpeg7:GivenName>
                     <mpeg7:FamilyName>Doe</mpeg7:FamilyName>
                  </mpeg7:Name>
               </UserInfo>
            </UserCharacteristic>
            <UserCharacteristic xsi:type="AudioPresentationPreferencesType">
               <VolumeControl>0.85</VolumeControl>
               <FrequencyEqualizer>
                  -10 -10 -10 -10 -10 -10 -10 -10 -10 -10
                  -10 -10 0   0   0   0  10  10  10  10
                  -10 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10
               </FrequencyEqualizer>
               <AudibleFrequencyRange>
                  <StartFrequency>20</StartFrequency>
                  <EndFrequency>20000</EndFrequency>
               </AudibleFrequencyRange>
               <Soundfield>
                  <ImpulseResponse href="http://www.sac.or.kr/concertHall/hallImp.wav">
                     <SamplingFrequency>44100</SamplingFrequency>
                     <BitsPerSample>16</BitsPerSample>
                     <NumOfChannels>1</NumOfChannels>
                  </ImpulseResponse>
               </Soundfield>
               <SoniferousSpeed>0.5</SoniferousSpeed>
            </UserCharacteristic>
         </User>
      </UsageEnvironmentProperty>
   </Description>
</DIA>
```

7.1.10 Device profile

Terminal characteristics are described in terms of:

- codec capabilities (encoding and decoding formats);
- device properties (user interaction support, e.g.: mouse, pen or other types of input devices; device class, e.g.: PC, PDA, Set-top box, etc.);

- input-output characteristics (display and audio output capabilities, e.g.: resolution, rendering format, bits/pixel, color capable, frequency ranges, output power, SNR, etc.).

Using DIA types and subtypes, the information mainly required by the Content Format Engine is:

- *Terminal*: specifies the capabilities and properties of a single terminal, that include coding and decoding capabilities, device properties and input-output capabilities;
- *CodecCapabilities*: specifies the format that a particular terminal is capable of encoding or decoding;
- *Display*: specifies the capabilities and properties of a single display;
- *AudioOutput*： specifies the capabilities and properties of a single audio output;
- *UserInteractionInput*: specifies the various types of User interaction input support that is available on a particular device;
- *DeviceClass*: makes possible to identify consumer devices such as PC and mobile phones from communication devices such as gateways and routers;
- *Storage*: specifies storage characteristics of the terminal, which does include input and output transfer rates, the size of the storage and whether the storage is writeable or not.

An example of device profile is the following:

```
<DIA>
   <Description xsi:type="UsageEnvironmentType">
      <UsageEnvironmentProperty xsi:type="TerminalsType">
         <Terminal>
            <TerminalCapability xsi:type="StoragesType">
               <Storage xsi:type="StorageType">
                  <StorageCharacteristic xsi:type="StorageCharacteristicsType"
                                 inputTransferRate="8" size="1200" writable="true"/>
               </Storage>
            </TerminalCapability>
            <TerminalCapability xsi:type="DeviceClassType">
               <DeviceClass href="urn:mpeg:mpeg21:2003:01-DIA-DeviceClassCS-NS:1">
                  <mpeg7:Name xml:lang="en">PC</mpeg7:Name>
               </DeviceClass>
            </TerminalCapability>
            <TerminalCapability xsi:type="DisplaysType">
               <Display id="primary_display">
                  <DisplayCapability xsi:type="DisplayCapabilityType">
                     <Mode>
                        <Resolution horizontal="720" vertical="480"/>
                     </Mode>
                  </DisplayCapability>
               </Display>
               <Display id="secondary_display">
                  <DisplayCapability xsi:type="DisplayCapabilityType">
                     <Mode>
                        <Resolution horizontal="176" vertical="144"/>
                     </Mode>
                  </DisplayCapability>
               </Display>
            </TerminalCapability>
            <TerminalCapability xsi:type="UserInteractionInputsType">
               <UserInteractionInput>
                  <UserInteractionInputSupport xsi:type="MicrophoneType"/>
               </UserInteractionInput>
               <UserInteractionInput>
                  <UserInteractionInputSupport xsi:type="KeyInputType">
                     <KeyInput href="urn:mpeg:mpeg21:2003:01-DIA-KeyInputCS-NS:1">
                        <mpeg7:Name xml:lang="en">PCKeyboard</mpeg7:Name>
                     </KeyInput>
                  </UserInteractionInputSupport>
               </UserInteractionInput>
               <UserInteractionInput>
                  <UserInteractionInputSupport xsi:type="MouseType">
                     <Mouse buttons="2" scrollwheel="true"/>
                  </UserInteractionInputSupport>
```

```
                </UserInteractionInput>
            </TerminalCapability>
            <TerminalCapability xsi:type="CodecCapabilitiesType">
                <Decoding xsi:type="AudioCapabilitiesType">
                    <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
                        <mpeg7:Name xml:lang="en">MP3</mpeg7:Name>
                    </Format>
                    <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
                        <mpeg7:Name xml:lang="en">AMR</mpeg7:Name>
                    </Format>
                </Decoding>
                <Decoding xsi:type="ImageCapabilitiesType">
                    <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
                        <mpeg7:Name xml:lang="en">JPEG</mpeg7:Name>
                    </Format>
                </Decoding>
                <Decoding xsi:type="VideoCapabilitiesType">
                    <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
                        <mpeg7:Name xml:lang="en">
                            MPEG-4 Visual Simple Profile @ Level 1
                        </mpeg7:Name>
                    </Format>
                </Decoding>
                <Encoding xsi:type="AudioCapabilitiesType">
                    <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
                        <mpeg7:Name xml:lang="en">AMR</mpeg7:Name>
                    </Format>
                </Encoding>
                <Encoding xsi:type="VideoCapabilitiesType">
                    <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
                        <mpeg7:Name xml:lang="en">
                            MPEG-4 Visual Simple Profile @ Level 1
                        </mpeg7:Name>
                    </Format>
                </Encoding>
            </TerminalCapability>
        </Terminal>
    </UsageEnvironmentProperty>
</Description>
</DIA>
```

7.1.11 Context profile

Context (network and natural environment) characteristics are described in terms of:

- capabilities (e.g.: capacity of the channel, minimum guaranteed bandwidth, etc.);
- conditions (e.g.: error rate, delay, available bandwidth, etc.);
- location and time;
- audio-visual (e.g.: audio noise levels, illumination conditions affecting a display, etc.).

Using DIA types and subtypes, the information mainly required by the Content Format Engine is:

- *Network*: describes a single network in terms of its static capabilities and time-varying conditions;
- *NaturalEnvironment*: describes a single natural environment in terms of the location and time of usage of a Digital Item, as well as audio-visual characteristics of the natural usage environment;
- *Location*: describes the location of the usage of a Digital Item;
- *Time*: describes the time of the usage of a Digital Item.

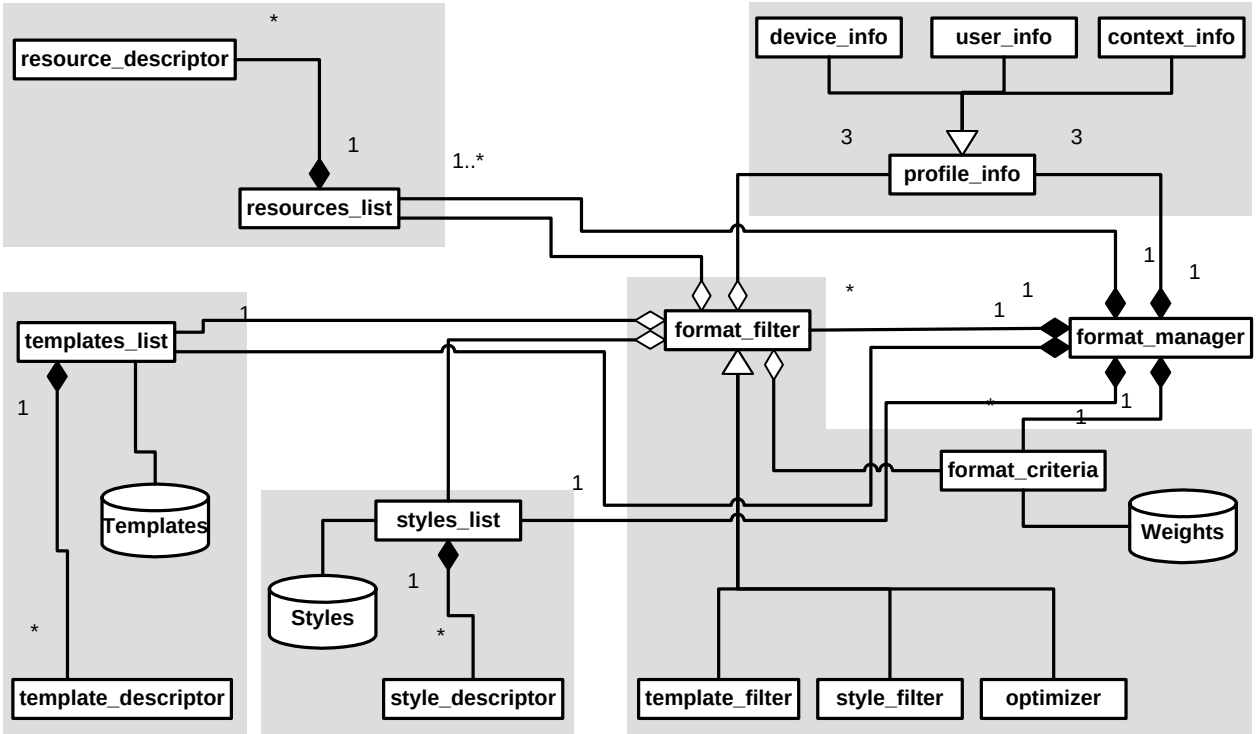
An example of context profile is the following:

```
<DIA>
    <Description xsi:type="UsageEnvironmentType">
        <UsageEnvironmentProperty xsi:type="NetworksType">
            <Network>
                <NetworkCharacteristic xsi:type="NetworkCapabilityType"
                            maxCapacity="384000" minGuaranteed="32000"/>
                <NetworkCharacteristic xsi:type="NetworkConditionType" duration="PT330N1000F">
```

```
                <AvailableBandwidth maximum="256000" average="80000"/>
                <Delay packetTwoWay="330" delayVariation="66"/>
                <Error packetLossRate="0.05"/>
            </NetworkCharacteristic>
        </Network>
    </UsageEnvironmentProperty>
    <UsageEnvironmentProperty xsi:type="NaturalEnvironmentsType">
        <NaturalEnvironment>
            <NaturalEnvironmentCharacteristic xsi:type="LocationType">
                <Location>
                    <mpeg7:GeographicPosition>
                        <mpeg7:Point longitude="135.75" latitude="35.00" altitude="10.00"/>
                    </mpeg7:GeographicPosition>
                    <mpeg7:Region>jp</mpeg7:Region>
                </Location>
            </NaturalEnvironmentCharacteristic>
        </NaturalEnvironment>
    </UsageEnvironmentProperty>
    <UsageEnvironmentProperty xsi:type="NaturalEnvironmentsType">
        <NaturalEnvironment>
            <NaturalEnvironmentCharacteristic xsi:type="TimeType">
                <Time>
                    <mpeg7:TimePoint>1998-07-10T15:22+01:00</mpeg7:TimePoint>
                </Time>
            </NaturalEnvironmentCharacteristic>
        </NaturalEnvironment>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

7.2 Module Design in terms of Classes

The class diagram for the Content Format Engine is depicted in the following figure:



The diagram is divided in five areas:

1. Resources management (*resource_descriptor* and *resources_list* classes);
2. Templates management (*template_descriptor* and *templates_list* classes);
3. Style-sheets management (*style_descriptor* and *styles_list* classes);

4. Profiles management (*device_info*, *user_info* and *context_info* classes, that specializes the *profile_info*);
5. Format management (*format_filter* - that generalizes *style_filter*, *template_filter* and *optimizer* - and *format_criteria* classes).

7.2.1 Format manager

The ***format_manager*** drives the whole process: this class receives reference to profiles (device, user and context) and resources; offers methods to browse and search templates and style-sheets, and to optimize them.

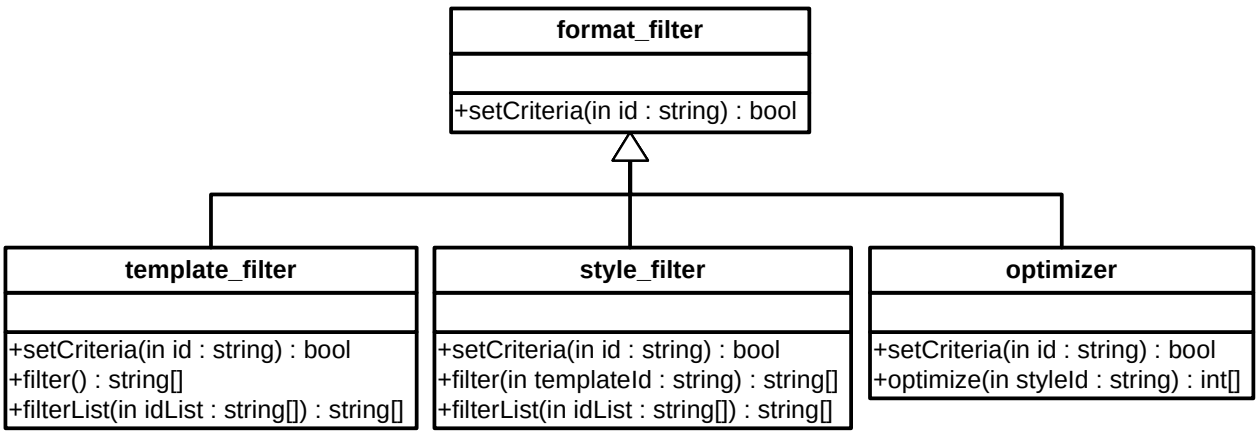
format_manager

+addResource(in uri : string) : bool
adds a new resource to the system
+addResourceDescriptor(in uri : string) : bool
adds a new XML resource descriptor to the system
+clearResources() : bool
empties the resources list
+addCriteria(in id : string, in type : string, in weights : float[], in meta : string) : bool
stores a new set of weights
+setCriteria(in id : string) : bool
set criteria for next filtering operation
+setDevice(in uri : string) : bool +setUser(in uri : string) : bool +setContext(in uri : string) : bool
set profiles stored in the given XML file
+filterTemplate() : string[]
returns a list of template IDs
+filterStyle(in templateId : string) : string[]
returns a list of style-sheet IDs
+optimize(in styleId : string) : int[]
returns the optimized values for parameters
+apply(in styleId : string, in values : int[]) : string
creates the SMIL document

The ***format_filter*** (attualmente non presente) class generalizes *template_filter*, *style_filter* and *optimizer classes*.

format_filter

+setCriteria(in id : string) : bool
sets criteria used for filtering



7.2.2 Resources management

The ***resource_descriptor*** class is used to describe digital resources involved in the formatting process.
Descriptors are not stored, but only created in memory during their formatting process.
The single-parameter constructor creates a resource_descriptor loading data from an XML file; the other one receives data as input parameters.

resource_descriptor

+resource_descriptor(in address : string)
constructor from XML file
+resource_descriptor(in id : string, in location : string, in type : string, in format : string, in properties : string[], in meta : string, in parentId : string, in relatedId : string, in category : string)
constructor
+getLocation() : string
getter method
+getType() : string
getter method
+getFormat() : string
getter method
+getProperties() : string[]
getter method
+getMeta() : string
getter method
+getParent() : string
getter method
+getRelated() : string
getter method
+getCategory() : string
getter method
+setLocation(in location : string) : bool
setter method
+setType(in type : string) : bool
setter method
+setFormat(in format : string) : bool
setter method
+setProperties(in properties : string[]) : bool
setter method
+setMeta(in meta : string) : bool
setter method
+setParent(in parentId : string) : bool
setter method
+setRelated(in relatedId : string) : bool
setter method
+setCategory(in category : string) : bool
setter method

The ***resources_list*** collects the descriptors of all the resources involved in the current formatting activity. It offers methods to browse and search resources and information about a single resource or aggregated data.

7.2.3 Templates management

The ***template_descriptor*** class is used to describe templates used in the Format Engine.
Descriptors are stored locally, but may refer to templates stored elsewhere.

template_descriptor

+template_descriptor(in id : string)
constructor
+getData() : string
getter method
+getLocation() : string +setLocation(in address : string) : bool
getter/setter method
+getMetadata() : string +setMetadata(in meta : string) : bool
getter/setter method
+getDevice() : string +setDevice(in device : string) : bool
getter/setter method
+getFormat() : string +setFormat(in outformat : string) : bool
getter/setter method
+getCategory() : string +setCategory(in category : string) : bool
getter/setter method
+getHierarchy() : string
returns the XML hierarchy of the resources within the template
+getRes() : int
returns the number of resources needed by the template
+getResByType(in type : string) : int
returns the number of resources of the given type needed by the template
+getTypes() : int
getter method

The ***templates_list*** collects all the template descriptors and offers methods to browse the list and to get individual or aggregate data.

templates_list

+templates_list(in uri : string)
constructor; needs the address of the descriptors directory
+load() : bool
fill the list with existing template descriptors
+add(in id : string, in location : string, in outformat : string, in category : string, in device : string, in meta : string) : bool
creates a new template descriptor
+addTemplate(in id : string, in xml : string, in outformat : string, in category : string, in device : string, in meta : string) : bool
creates and stores a new template and its descriptor
+getData(in id : string) : string
getter method
+getLocation(in id : string) : string
getter method
+getMetadata(in id : string) : string
getter method
+getDevice(in id : string) : string
getter method
+getFormat(in id : string) : string

getter method
+getCategory(in id : string) : string
getter method
+getHierarchy(in id : string) : string
returns the XML hierarchy of the resources within the template
+getRes(in id : string) : int
returns the number of resources needed by the template
+getResByType(in id : string) : int
returns the number of resources of the given type needed by the template
+getType(in id : string) : int
returns the number of different resource types needed by the template

7.2.4 Templates filtering

The ***template_filter*** class sorts templates contained in *templates_list*, according a *format_criteria*.

template_filter

+setCriteria(in id : string) : bool
sets criteria used for filtering
+filter() : string[]
returns IDs of selected templates
+filterList(in idList : string[]) : string[]
filters a list of already filtered templates

7.2.5 Style-sheets management

The ***style_descriptor*** class is used to describe style-sheets used in the Format Engine. Each style-sheet is univoquely linked to a template with its ID. Descriptors are stored locally, but may refer to style-sheets stored elsewhere.

style_descriptor

+style_descriptor(in id : string)
constructor
+getData() : string
getter method
+getLocation() : string +setLocation(in address : string) : bool
getter/setter method
+getMetadata() : string +setMetadata(in meta : string) : bool
getter/setter method
+getDevice() : string +setDevice(in device : string) : bool
getter/setter method
+getTemplate() : string +setTemplate(in template : string) : bool
getter/setter method
+getParamNum() : int
returns the number of parameters defined within the style-sheet
+getParams() : string[]
returns the list of parameters defined within the style-sheet

The ***styles_list*** collects all the style-sheet descriptors and offers methods to browse the list and to get individual or aggregated data.

styles_list

+styles_list(in uri : string)
constructor; needs the address of the descriptors directory
+load() : bool
fill the list with existing style-sheet descriptors
+add(in id : string, in location : string, in device : string, in metadata : string, in template : string) : bool
creates a new style descriptor
+addStyle(in xml : string, in id : string, in device : string, in metadata : string, in template : string) : bool
creates and stores a new style-sheet and its descriptor
+getData(in id : string) : string
getter method
+getLocation(in id : string) : string
getter method
+getMetadata(in id : string) : string
getter method
+getDevice(in id : string) : string
getter method
+getTemplate(in id : string) : string
getter method
+getParamNum(in id : string) : int
returns the number of parameters defined within the style-sheet
+getParams(in id : string) : string[]
returns the list of parameters defined within the style-sheet

7.2.6 Style-sheets filtering

The ***style_filter*** class sorts style-sheets contained in *styles_list*, according a *format_criteria*.

style_filter

+setCriteria(in id : string) : bool
sets criteria used for filtering
+filter(in templateId: string) : string[]
returns IDs of selected style-sheets for the given template
+filterList(in idList : string[]) : string[]
filters a list of already filtered style-sheets

7.2.7 Style-sheet optimization

The ***optimizer*** class determines optimized values for parameters contained in the style-sheet, according to a *format_criteria*.

optimizer

+setCriteria(in id : string) : bool
sets criteria used for optimization
+optimize(in styleId : string) : int[]
optimize the selected style-sheet

7.2.8 Criteria

The ***format_criteria*** class manages weights for templates filtering, style-sheets filtering and optimization.

format_criteria

+addWeights(in id : string, in meta : string, in type : string, in weights : int[]) : bool

stores a new set of weights with its related information
+getWeights(in id : string) : int[] +setWeights(in id : string, in weights : int[]) : bool
getter/setter method
+getType(in id : string) : string +setType(in type : string, in id : string) : bool
getter/setter method
+getMetadata(in id : string) : string +setMetadata(in meta : string, in id : string) : bool
getter/setter method

7.3 Usage of AXMEDIS Configuration Manager

7.3.1 Modules used in the AxEditor configuration

In the AxEditor configuration (/Applications/axeditor/bin/win32/editor-configuration.xml), the following module defines parameters used by the formatting functionalities:

```
<Module category="" id="FORMATTING_CONF">
  <Parameter name="TemplDescrLocation" type="string">
    c:/
  </Parameter>
  <Parameter name="DefaultTemplLocation" type="string">
    c:/
  </Parameter>
  <Parameter name="StyleDescrLocation" type="string">
    c:/
  </Parameter>
  <Parameter name="DefaultStyleLocation" type="string">
    c:/
  </Parameter>
  <Parameter name="ToolsLocation" type="string">
    Framework/doc/test/contentformatter/files/tools/
  </Parameter>
</Module>
```

where:

- TemplDescrLocation defines the system directory where template descriptors are stored;
- DefaultTemplLocation defines the directory where new templates are created by the AxEditor;
- StyleDescrLocation defines the system directory where style-sheet descriptors are stored;
- DefaultStyleLocation defines the directory where new style-sheets are created by the AxEditor;
- ToolsLocation defines the directory that contains XSLT style-sheets for creating templates and style-sheets (*gentemplates.xsl* and *genstyle.xsl*, for instance).

7.3.2 Modules used in the RuleEditor configuration

In the RuleEditor configuration (\Applications\axeditor\bin\win32\configuration.xml), the following module defines parameters used by the formatting functionalities:

```
<Module category="" id="FORMATTING_CONF">
  <Parameter name="TemplDescrLocation" type="string">
    Framework/doc/test/contentformatter/files/templateDescriptor
  </Parameter>
  <Parameter name="StyleDescrLocation" type="string">
    Framework/doc/test/contentformatter/files/styleDescriptor
  </Parameter>
</Module>
```

7.4 Examples of usage

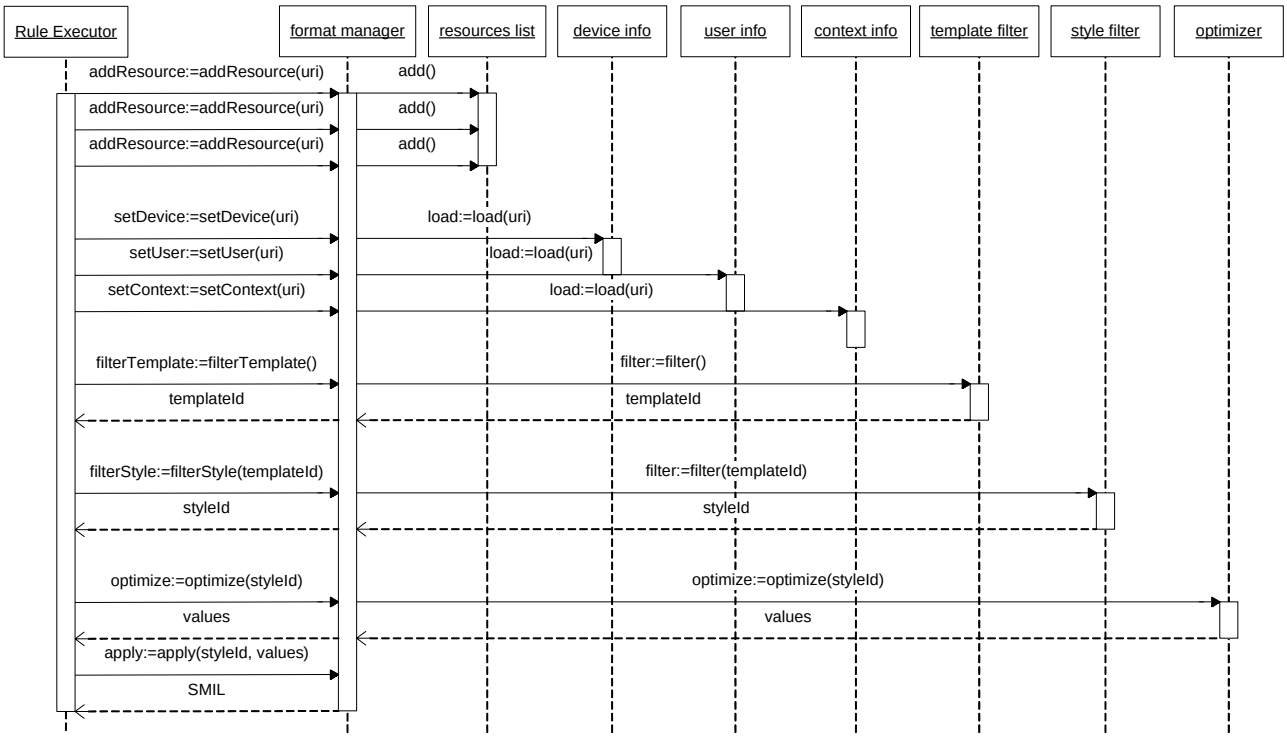
7.4.1 Formatting

The format_manager is used to manage the whole formatting process that starts with setting the inputs and ends with the SMIL production. The following example (where the *path* parameter is supposed to be the Framework/doc/test/contentformatter/ directory of the AXMEDIS SVN repository) describes a possible interaction with a Rule Executor; a layout is created for the resources associated with the descriptors contained in a given directory:

```
int test_format(string path, bool optimize = false)
{
  FormatManager fm;

  // load templates
  TemplateList* myTL = fm.getTemplateList();
  myTL->setDescrDir(path + "files/templateDescriptor");
  if (!myTL->loadDescriptors(path + "xsd/template_descriptor.xsd"))
  {
    // ERROR WHILE LOADING TEMPLATES
    return -1;
  }

  // load style-sheets
  StyleList* mySL = fm.getStyleList();
  mySL->setDescrDir(path + "files/styleDescriptor");
  if (!mySL->loadDescriptors())
  {
    // ERROR WHILE LOADING STYLES
    return -1;
  }

  // load profiles
  if (!fm.loadDeviceInfo(path + "files/profiles/device2.xml")) {
  {
    // ERROR WHILE LOADING PROFILES
    return -1;
  }

  // load resources contained in files/resourceDescriptor/play-stop/
  if (!fm.loadResourceDescriptors(path + files/resourceDescriptor/play-stop))
  {
    // ERROR WHILE LOADING RESOURCES
    return -1;
  }

  // filtering
  string idTemplate = fm.filterTemplate().front().getId();
  string idStyle = fm.filterStyle(idTemplate).front().getId();

  // (optimize and) create output
  xercesc::DOMDocument* theResult;
  if (!optimize)
    theResult = fm.saveSMIL(idTemplate, idStyle, "files/output/output.smi");
  else
    theResult = fm.saveOptimizedSMIL(idTemplate, idStyle,
                                     "files/output/output.smi", 800, 600);
  if (theResult != NULL)
    return 0;
```

```
  else
    return -1;
}
```

The following sequence diagram illustrates the whole interaction and roles played by different objects:



7.4.2 Creation of templates and style-sheets

The following example (that uses the XSLT style-sheets *gentemplate.xsl* and *genstyle.xsl* located in the Framework/doc/test/contentformatter/files/tools/ directory of the AXMEDIS SVN repository) tests the creation of new template and style-sheet based on an existing SMIL file:

```
int test_createTemplateAndStyleFromSMIL(string path)
{
  FormatManager fm;

  // load templates
  TemplateList* myTL = fm.getTemplateList();
  myTL->setDescrDir(path + "files/templateDescriptor");
  if (!myTL->loadDescriptors(path + "files/xsd/template_descriptor.xsd"))
  {
    // ERROR WHILE LOADING TEMPLATES
    return -1;
  }

  // load style-sheets
  StyleList* mySL = fm.getStyleList();
  mySL->setDescrDir(path + "files/styleDescriptor");
  if (!mySL->loadDescriptors())
  {
    // ERROR WHILE LOADING STYLES." << endl;
    return -1;
  }

  if (!fm.createDescriptorsFromSMIL(path + "files/tools/test.smi",
                                    path + "files/tools/gentemplate.xsl",
                                    path + "files/tools/genstyle.xsl",
                                    path + "files/templates/test_template.smi",
```

```
                                        path + "files/styles/test.xsl",
                                        "test")) {
    // ERROR WHILE CREATING FILES
    return -1;
  }
  return 0;
}
```

7.5 Formal description of algorithm

7.5.1 Genetic Algorithm

Genetic Algorithms are inspired by Darwin's theory of evolution and use an evolutionary process to find solutions of a problem.

Algorithm begins with a set of solutions (represented by “chromosomes”) called “population”. Solutions from one population are taken and used to form a new population, using crossover and mutation mechanisms. This is motivated by a hope, that the new population will be better than the old one. Solutions are selected to form new solutions (“offspring”) according to their fitness: the more suitable they are the more chances they have to reproduce. This is repeated until some condition (e.g.: number of populations, improvement of the best solution, etc.) is satisfied.

GAs evolve according to the following pseudo-code description:

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [New population] Create a new population by repeating following steps until the new population is complete:
   3.1. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   3.2. [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
   3.3. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
   3.4. [Accepting] Place new offspring in the new population
4. [Replace] Use new generated population for a further run of the algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 2

7.6 Formal description of format

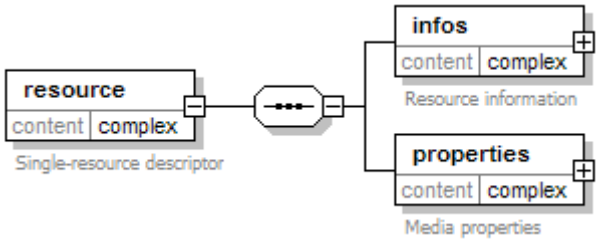
7.6.1 Formal description of resource descriptor

The following XML Schema defines the resource descriptor (resource_descriptor.xsd).

Elements	Simple types
resource	**applicationSubType**
	audioSubType
	categoryType
	imageSubType
	mimeType
	textSubType
	videoSubType

element **resource**

diagram

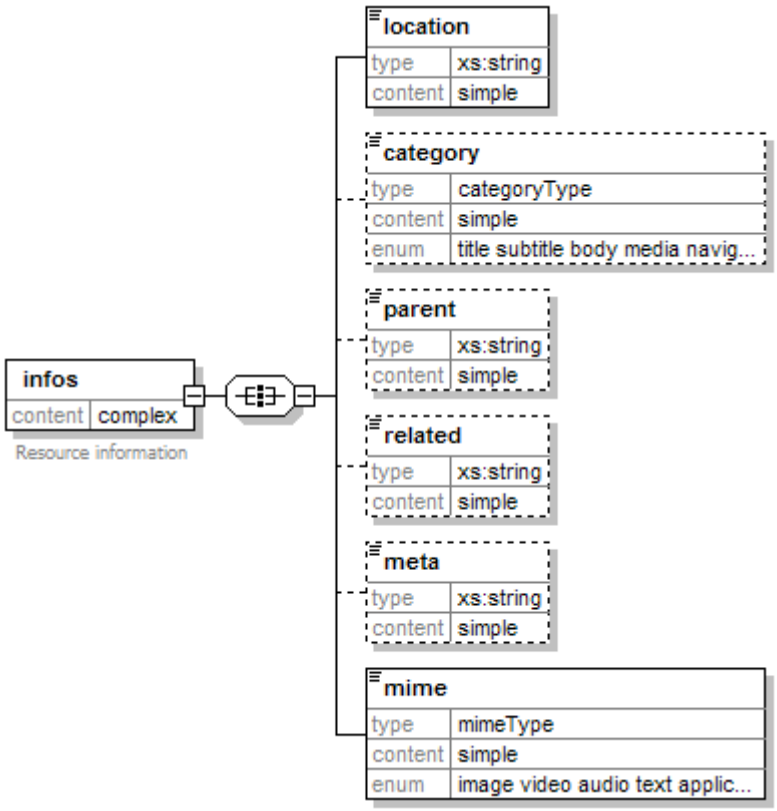


children **infos properties**

annotation documentation Single-resource descriptor

element **resource/infos**

diagram

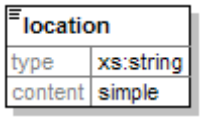


children **location category parent related meta mime**

annotation documentation Resource information

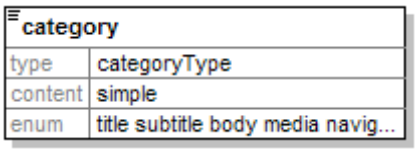
element **resource/infos/location**

diagram



element **resource/infos/category**

diagram

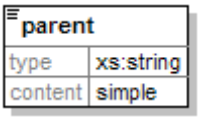


facets

enumeration	title
enumeration	subtitle
enumeration	body
enumeration	media
enumeration	navigation
enumeration	control
enumeration	content
enumeration	footer
enumeration	button

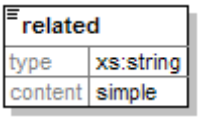
element **resource/infos/parent**

diagram



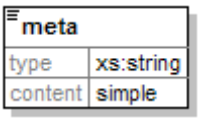
element **resource/infos/related**

diagram



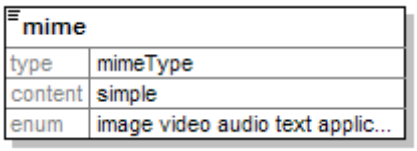
element **resource/infos/meta**

diagram



element **resource/infos/mime**

diagram

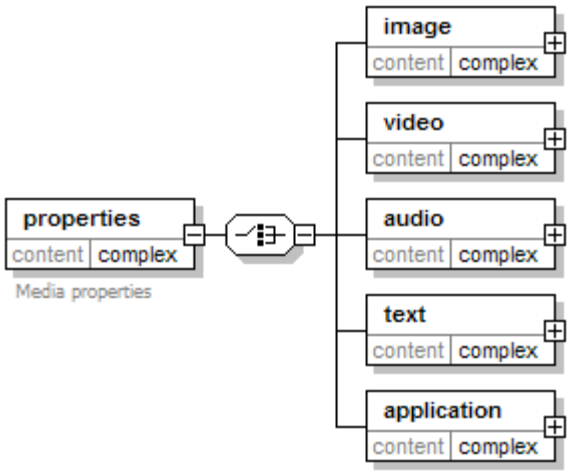


facets

enumeration	image
enumeration	video
enumeration	audio
enumeration	text
enumeration	application

element **resource/properties**

diagram

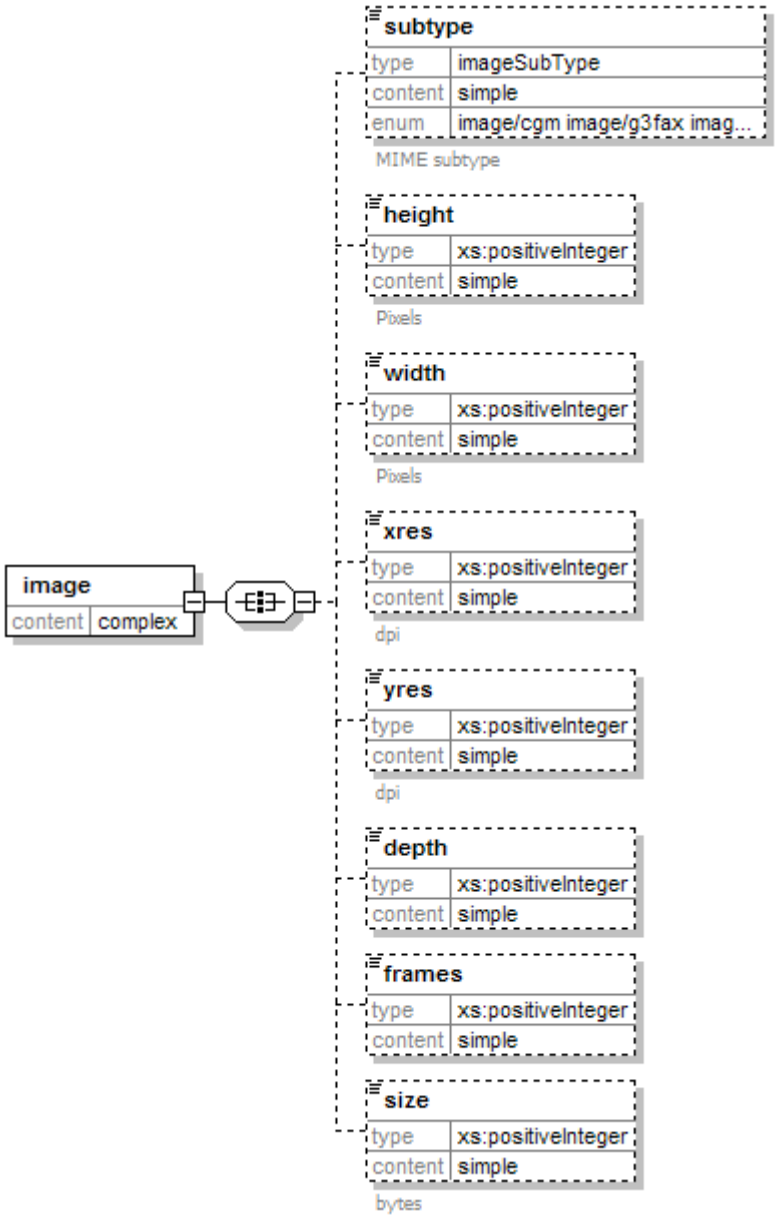


children **image video audio text application**

annotation documentation Media properties

element **resource/properties/image**

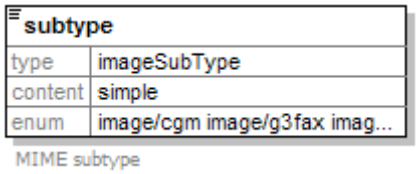
diagram



children **subtype height width xres yres depth frames size**

element **resource/properties/image/subtype**

diagram



facets

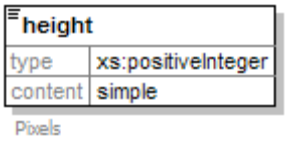
enumeration	image/cgm
enumeration	image/g3fax
enumeration	image/gif
enumeration	image/ief
enumeration	image/jpeg
enumeration	image/naplps
enumeration	image/png

enumeration image/prs.btif
enumeration image/prs.pti
enumeration image/tiff
enumeration image/vnd.cns.inf2
enumeration image/vnd.dwg
enumeration image/vnd.dxf
enumeration image/vnd.fastbidsheet
enumeration image/vnd.fpx
enumeration image/vnd.fst
enumeration image/vnd.fujixerox.edmics-mmr
enumeration image/vnd.fujixerox.edmics-rlc
enumeration image/vnd.mix
enumeration image/vnd.net-fpx
enumeration image/vnd.svf
enumeration image/vnd.wap.wbmp
enumeration image/vnd.xiff

annotation documentation MIME subtype

element **resource/properties/image/height**

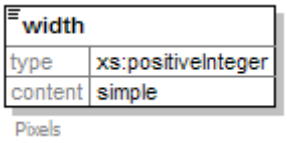
diagram



annotation documentation Pixels

element **resource/properties/image/width**

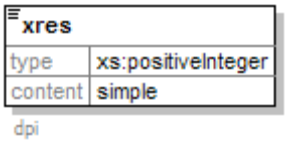
diagram



annotation documentation Pixels

element **resource/properties/image/xres**

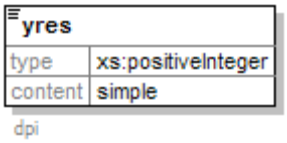
diagram



annotation documentation dpi

element **resource/properties/image/yres**

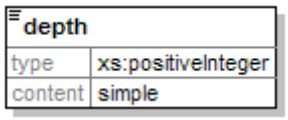
diagram



annotation documentation dpi

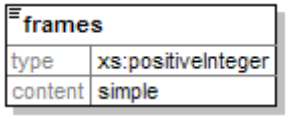
element **resource/properties/image/depth**

diagram



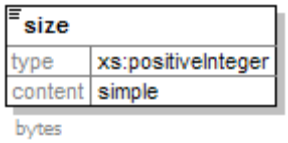
element **resource/properties/image/frames**

diagram



element **resource/properties/image/size**

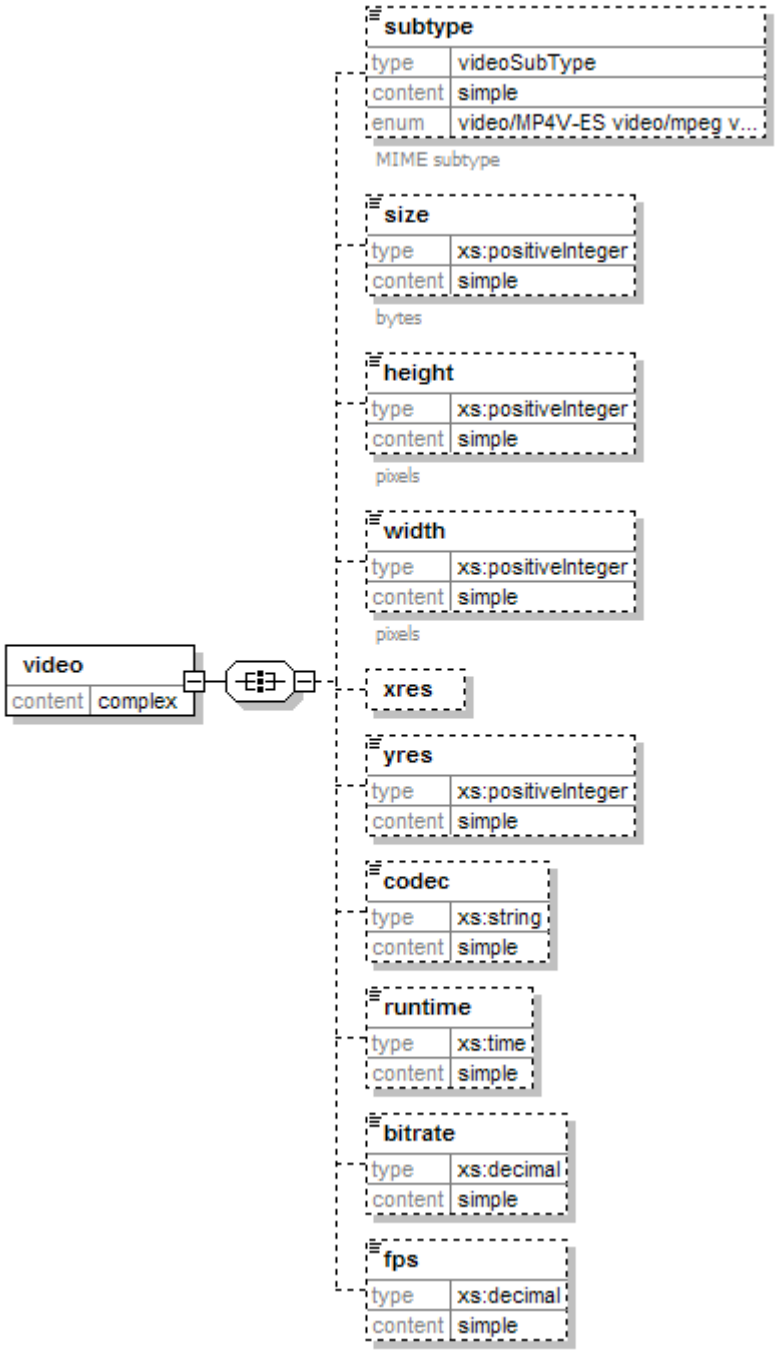
diagram



annotation documentation bytes

element **resource/properties/video**

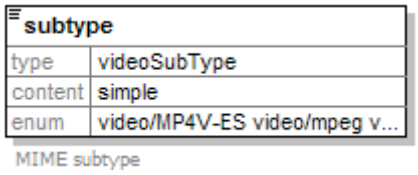
diagram



children **subtype size height width xres yres codec runtime bitrate fps**

element **resource/properties/video/subtype**

diagram

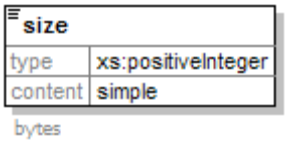


facets
enumeration video/MP4V-ES
enumeration video/mpeg

enumeration video/parityfec
enumeration video/pointer
enumeration video/quicktime
anoenumeration video/vnd.fvt
on
enumeration video/vnd.motorola.video/
enumeration video/vnd.motorola.videop
enumeration video/vnd.mpegurl
enumeration video/vnd.nokia.interleaved-multimedia
enumeration video/vnd.vivo
annotation documentation MIME subtype

element **resource/properties/video/size**

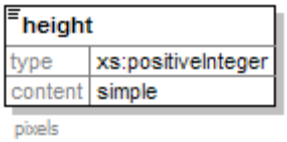
diagram



annotation documentation bytes

element **resource/properties/video/height**

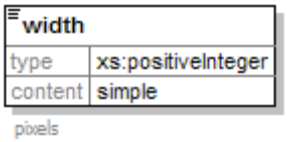
diagram



annotation documentation pixels

element **resource/properties/video/width**

diagram



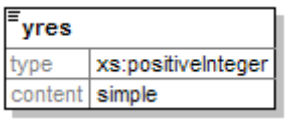
annotation documentation pixels

element **resource/properties/video/xres**

diagram

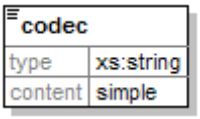
element **resource/properties/video/yres**

diagram



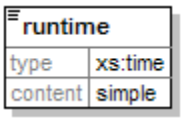
element **resource/properties/video/codec**

diagram



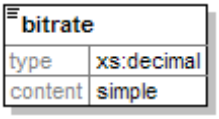
element **resource/properties/video/runtime**

diagram



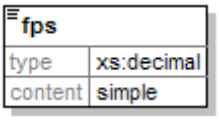
element **resource/properties/video/bitrate**

diagram



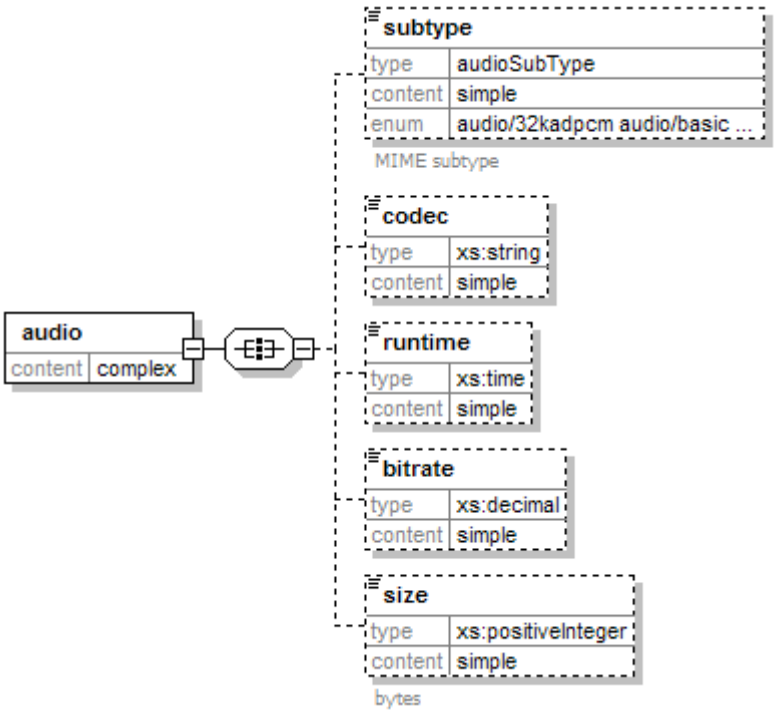
element **resource/properties/video/fps**

diagram



element **resource/properties/audio**

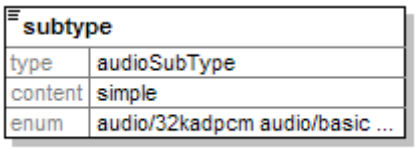
diagram



children **subtype codec runtime bitrate size**

element **resource/properties/audio/subtype**

diagram



facets

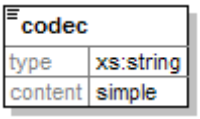
enumeration	audio/32kadpcm
enumeration	audio/basic
enumeration	audio/DAT12
enumeration	audio/G.722.1
enumeration	audio/L16
enumeration	audio/L20
enumeration	audio/L24
enumeration	audio/MP4A-LATM
enumeration	audio/mpa-robust
enumeration	audio/mpeg
enumeration	audio/parityfec
enumeration	audio/prs.sid
enumeration	audio/telephone-event
enumeration	audio/tone
enumeration	audio/vnd.cisco.nse
enumeration	audio/vnd.cns.anp1
enumeration	audio/vnd.cns.inf1
enumeration	audio/vnd.digital-winds
enumeration	audio/vnd.everad.plj
enumeration	audio/vnd.lucent.voice
enumeration	audio/vnd.nortel.vbk
enumeration	audio/vnd.nuera.ecelp4800
enumeration	audio/vnd.nuera.ecelp7470
enumeration	audio/vnd.nuera.ecelp9600
enumeration	audio/vnd.octel.sbc
enumeration	audio/vnd.qcelp
enumeration	audio/vnd.rhetorex.32kadpcm
enumeration	audio/vnd.vmx.cvsd

annotation

documentation	MIME subtype

element **resource/properties/audio/codec**

diagram

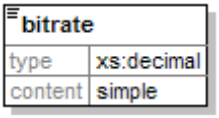


element **resource/properties/audio/runtime**

diagram

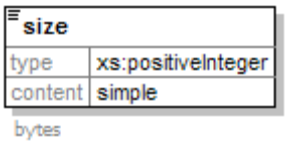
element **resource/properties/audio/bitrate**

diagram



element **resource/properties/audio/size**

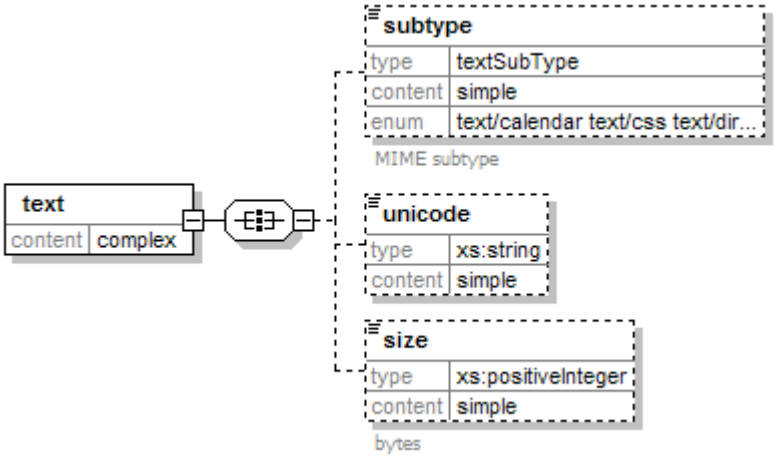
diagram



annotation documentation bytes

element **resource/properties/text**

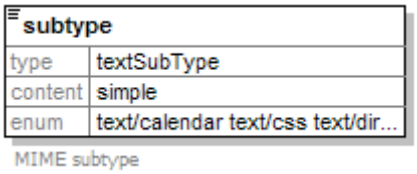
diagram



children **subtype** **unicode** **size**

element **resource/properties/text/subtype**

diagram



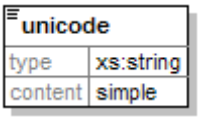
facets

enumeration	text/calendar
enumeration	text/css
enumeration	text/directory
enumeration	text/enriched
enumeration	text/html
enumeration	text/parityfec
enumeration	text/plain
enumeration	text/prs.lines.tag
enumeration	text/rfc822-headers
enumeration	text/richtext
enumeration	text/rtf
enumeration	text/sgml
enumeration	text/t140
enumeration	text/tab-separated-values
enumeration	text/uri-list
enumeration	text/vnd.curl
enumeration	text/vnd.fly
enumeration	text/vnd.fmi.flexstor
enumeration	text/vnd.in3d.spot
enumeration	text/vnd.IPTC.NewsML
enumeration	text/vnd.IPTC.NITF
enumeration	text/vnd.latex-z
enumeration	text/vnd.ms-mediapackage
enumeration	text/vnd.wap.si
enumeration	text/vnd.wap.sl
enumeration	text/xml

annotation
enumeration text/xml-external-parsed-entity
documentation MIME subtype

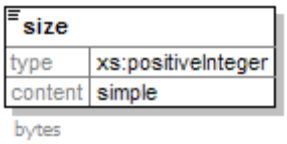
element **resource/properties/text/unicode**

diagram



element **resource/properties/text/size**

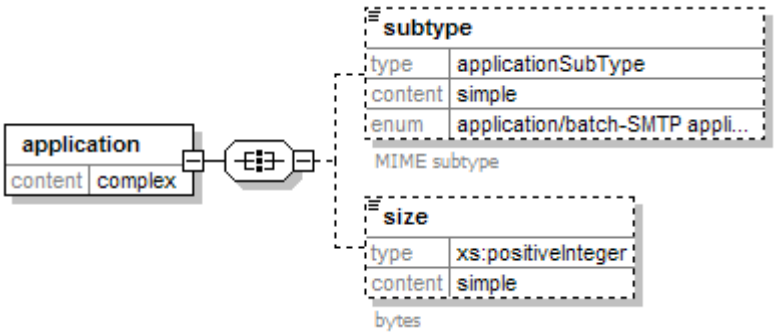
diagram



annotation
documentation bytes

element **resource/properties/application**

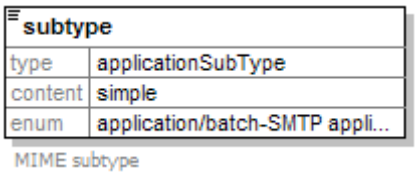
diagram



children **subtype** **size**

element **resource/properties/application/subtype**

diagram



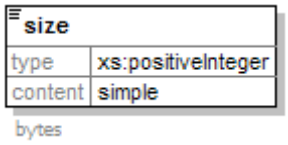
facets
enumeration application/batch-SMTP
enumeration application/beep+xml
enumeration application/cals-1840
enumeration application/dvcs
enumeration application/EDI-Consent
enumeration application/EDIFACT
enumeration application/EDI-X12
enumeration application/font-tdpfr
enumeration application/http
enumeration application/index
enumeration application/index.cmd
enumeration application/index.obj
enumeration application/index.response
enumeration application/index.vnd
enumeration application/iotp
enumeration application/ipp
enumeration application/isup
enumeration application/marc

enumeration application/msword
enumeration application/news-message-id
enumeration application/news-transmission
enumeration application/ocsp-request
enumeration application/ocsp-response
enumeration application/octet-stream
enumeration application/oda
enumeration application/parityfec
enumeration application/pdf
enumeration application/pgp-encrypted
enumeration application/pgp-keys
enumeration application/pgp-signature
enumeration application/pkix-cert
enumeration application/pkixcmp
enumeration application/pkix-crl
enumeration application/postscript
enumeration application/qsig
enumeration application/sdp
enumeration application/sgml
enumeration application/sieve
enumeration application/timestamp-query
enumeration application/timestamp-reply
enumeration application/vemmi
enumeration application/whoispp-query
enumeration application/whoispp-response
enumeration application/x400-bp
enumeration application/xml
enumeration application/xml-dtd
enumeration application/xml-external-parsed-entity
enumeration application/zip

annotation documentation MIME subtype

element **resource/properties/application/size**

diagram



annotation documentation bytes

simpleType **applicationSubType**

facets enumeration application/batch-SMTP
enumeration application/beep+xml
enumeration application/cals-1840
enumeration application/dvcs
enumeration application/EDI-Consent
enumeration application/EDIFACT
enumeration application/EDI-X12
enumeration application/font-tdpfr
enumeration application/http
enumeration application/index
enumeration application/index.cmd
enumeration application/index.obj
enumeration application/index.response
enumeration application/index.vnd
enumeration application/iotp
enumeration application/ipp
enumeration application/isup
enumeration application/marc
enumeration application/msword
enumeration application/news-message-id
enumeration application/news-transmission
enumeration application/ocsp-request
enumeration application/ocsp-response
enumeration application/octet-stream
enumeration application/oda
enumeration application/parityfec
enumeration application/pdf

enumeration application/pgp-encrypted
enumeration application/pgp-keys
enumeration application/pgp-signature
enumeration application/pkix-cert
enumeration application/pkixcmp
enumeration application/pkix-crl
enumeration application/postscript
enumeration application/qsig
enumeration application/sdp
enumeration application/sgml
enumeration application/sieve
enumeration application/timestamp-query
enumeration application/timestamp-reply
enumeration application/vemmi
enumeration application/whoispp-query
enumeration application/whoispp-response
enumeration application/x400-bp
enumeration application/xml
enumeration application/xml-dtd
enumeration application/xml-external-parsed-entity
enumeration application/zip

annotation documentation MIME subtypes for application media type

simpleType **audioSubType**

facets enumeration audio/32kadpcm
enumeration audio/basic
enumeration audio/DAT12
enumeration audio/G.722.1
enumeration audio/L16
enumeration audio/L20
enumeration audio/L24
enumeration audio/MP4A-LATM
enumeration audio/mpa-robust
enumeration audio/mpeg
enumeration audio/parityfec
enumeration audio/prs.sid
enumeration audio/telephone-event
enumeration audio/tone
enumeration audio/vnd.cisco.nse
enumeration audio/vnd.cns.anp1
enumeration audio/vnd.cns.inf1
enumeration audio/vnd.digital-winds
enumeration audio/vnd.everad.plj
enumeration audio/vnd.lucent.voice
enumeration audio/vnd.nortel.vbk
enumeration audio/vnd.nuera.ecelp4800
enumeration audio/vnd.nuera.ecelp7470
enumeration audio/vnd.nuera.ecelp9600
enumeration audio/vnd.octel.sbc
enumeration audio/vnd.qcelp
enumeration audio/vnd.rhetorex.32kadpcm
enumeration audio/vnd.vmx.cvsd

annotation documentation MIME subtypes for audio media type

simpleType **categoryType**

facets enumeration title
enumeration subtitle
enumeration body
enumeration media
enumeration navigation
enumeration control
enumeration content
enumeration footer
enumeration button

annotation documentation Information about the resource category

simpleType **imageSubType**

facets enumeration image/cgm

enumeration image/g3fax
enumeration image/gif
enumeration image/ief
enumeration image/jpeg
enumeration image/naplps
enumeration image/png
enumeration image/prs.btif
enumeration image/prs.pti
enumeration image/tiff
enumeration image/vnd.cns.inf2
enumeration image/vnd.dwg
enumeration image/vnd.dxf
enumeration image/vnd.fastbidsheet
enumeration image/vnd.fpx
enumeration image/vnd.fst
enumeration image/vnd.fujixerox.edmics-mmr
enumeration image/vnd.fujixerox.edmics-rlc
enumeration image/vnd.mix
enumeration image/vnd.net-fpx
enumeration image/vnd.svf
enumeration image/vnd.wap.wbmp
enumeration image/vnd.xiff

annotation documentation MIME subtypes for image media type

simpleType **mimeType**

facets
enumeration image
enumeration video
enumeration audio
enumeration text
enumeration application

simpleType **textSubType**

facets
enumeration text/calendar
enumeration text/css
enumeration text/directory
enumeration text/enriched
enumeration text/html
enumeration text/parityfec
enumeration text/plain
enumeration text/prs.lines.tag
enumeration text/rfc822-headers
enumeration text/richtext
enumeration text/rtf
enumeration text/sgml
enumeration text/t140
enumeration text/tab-separated-values
enumeration text/uri-list
enumeration text/vnd.curl
enumeration text/vnd.fly
enumeration text/vnd.fmi.flexstor
enumeration text/vnd.in3d.spot
enumeration text/vnd.IPTC.NewsML
enumeration text/vnd.IPTC.NITF
enumeration text/vnd.latex-z
enumeration text/vnd.ms-mediapackage
enumeration text/vnd.wap.si
enumeration text/vnd.wap.sl
enumeration text/xml
enumeration text/xml-external-parsed-entity

annotation documentation MIME subtypes for text media type

simpleType **videoSubType**

facets
enumeration video/MP4V-ES
enumeration video/mpeg
enumeration video/parityfec
enumeration video/pointer
enumeration video/quicktime
enumeration video/vnd.fvt
enumeration video/vnd.motorola.video/

enumeration video/vnd.motorola.videop
enumeration video/vnd.mpegurl
enumeration video/vnd.nokia.interleaved-multimedia
enumeration video/vnd.vivo
annotation documentation MIME subtypes for video media type

7.6.2 Formal description of template descriptor

The following XML Schema defines the template descriptor (template_descriptor.xsd).

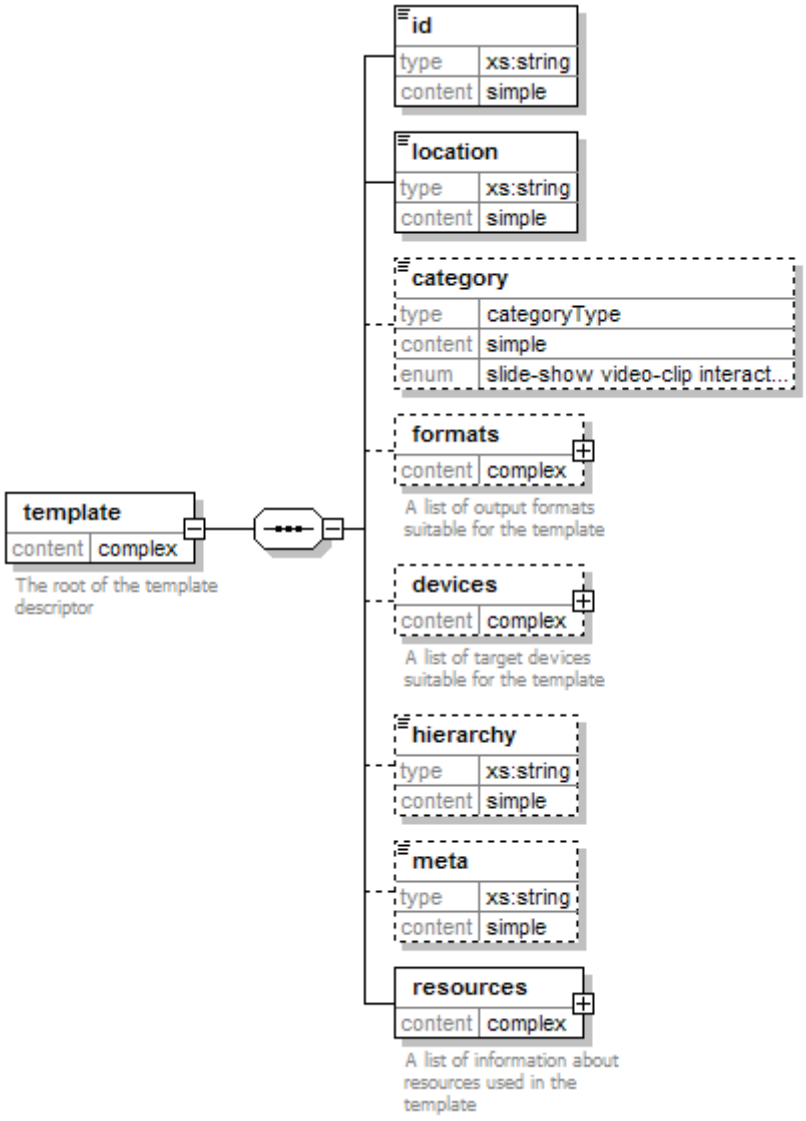
Schema **template_descriptor.xsd**

Elements	Complex types	Simple types
template	**infoType**	**categoryType**
	usageType	**mediaType**
		outformatType
		resourceCategoryType

Simple types
deviceType

element **template**

diagram

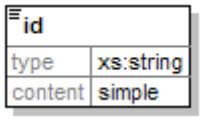


children **id location category formats devices hierarchy meta resources**

annotation documentation The root of the template descriptor

element **template/id**

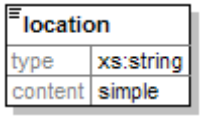
diagram



type **xs:string**

element **template/location**

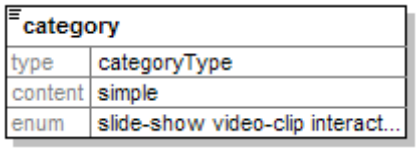
diagram



type **xs:string**

element **template/category**

diagram

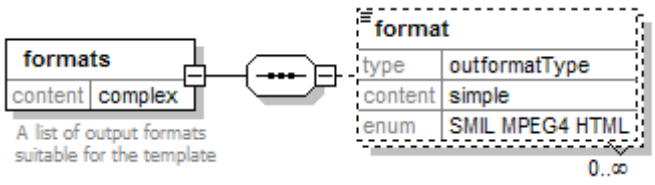


type **categoryType**

facets
- enumeration slide-show
- enumeration video-clip
- enumeration interactive-music
- enumeration interactive-video
- enumeration hypertext
- enumeration audio-karaoke
- enumeration video-karaoke
- enumeration e-book
- enumeration e-learning
- enumeration tutorial
- enumeration unknown

element **template/formats**

diagram

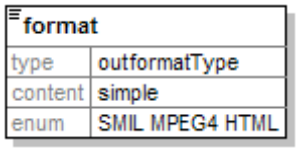


children **format**

annotation documentation A list of output formats suitable for the template

element **template/formats/format**

diagram

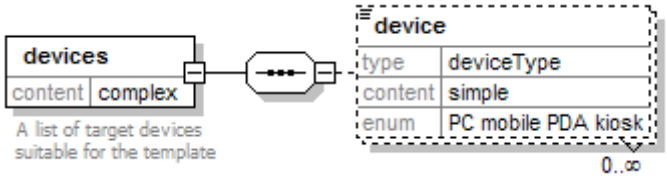


type **outformatType**

facets
- enumeration SMIL
- enumeration MPEG4
- enumeration HTML

element **template/devices**

diagram

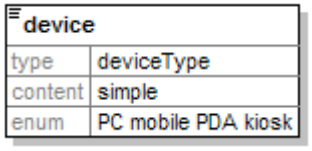


children **device**

annotation
- documentation A list of target devices suitable for the template

element **template/devices/device**

diagram

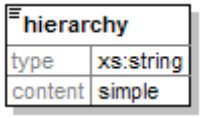


type **deviceType**

facets
- enumeration PC
- enumeration mobile
- enumeration PDA
- enumeration kiosk

element **template/hierarchy**

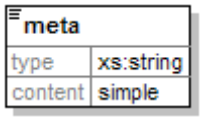
diagram



type **xs:string**

element **template/meta**

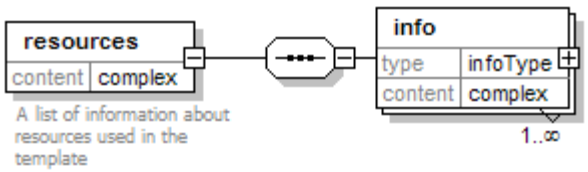
diagram



type **xs:string**

element **template/resources**

diagram

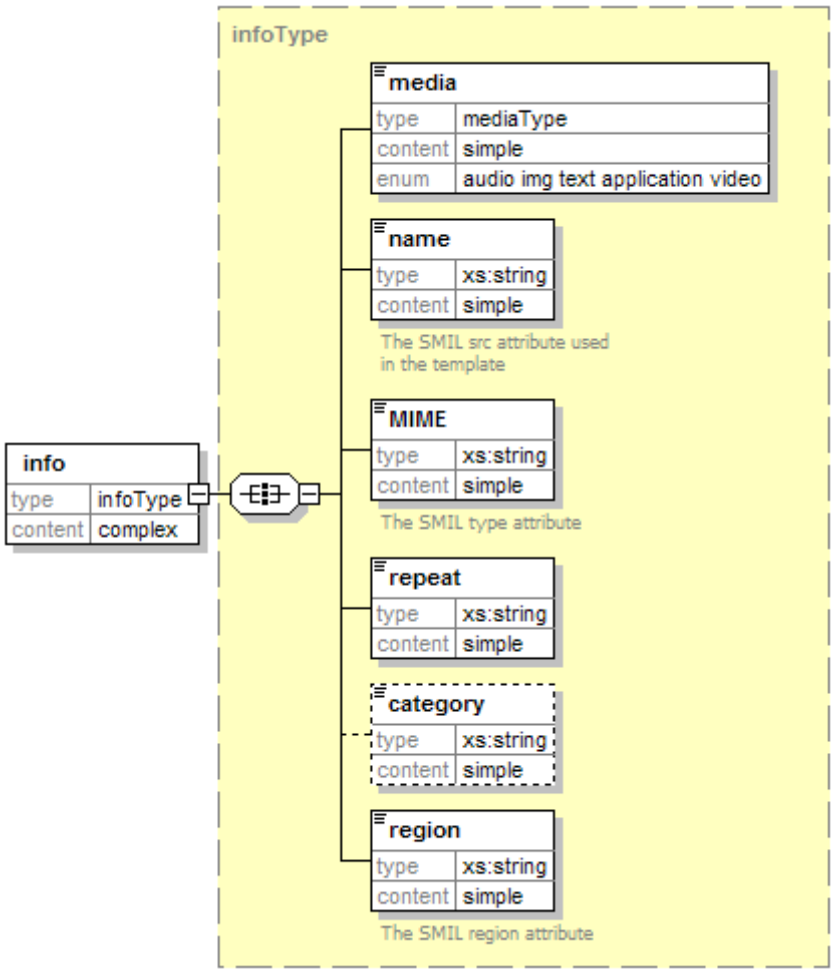


children **info**

annotation
- documentation A list of information about resources used in the template

element **template/resources/info**

diagram

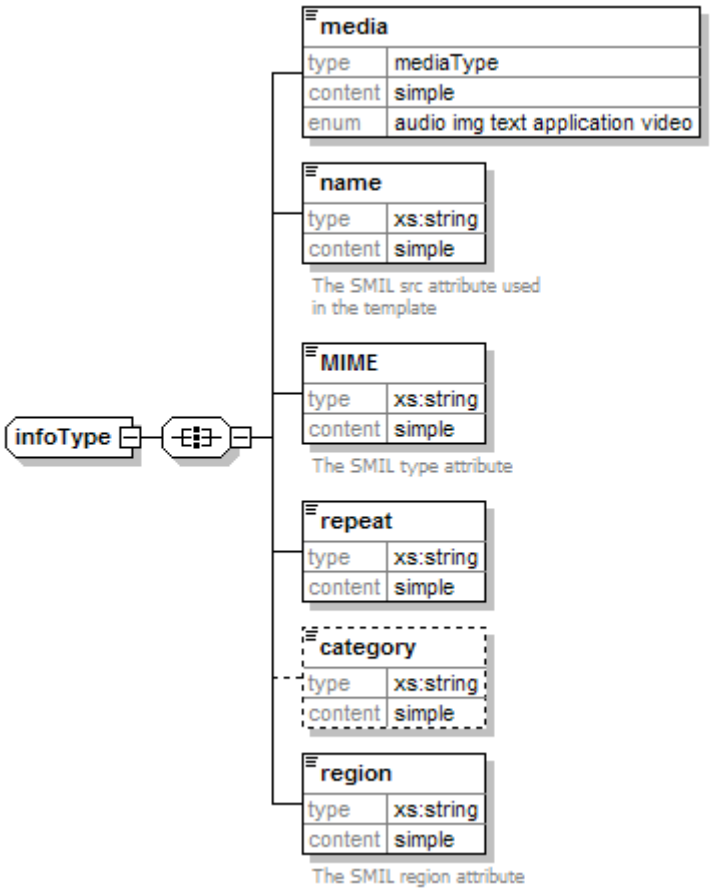


type **infoType**

children **media** **name** **MIME** **repeat** **category** **region**

complexType **infoType**

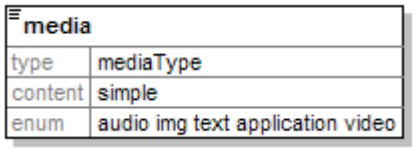
diagram



children **media** **name** **MIME** **repeat** **category** **region**

element **infoType/media**

diagram

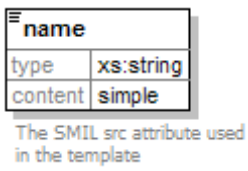


type **mediaType**

facets
- enumeration audio
- enumeration img
- enumeration text
- enumeration application
- enumeration video

element **infoType/name**

diagram

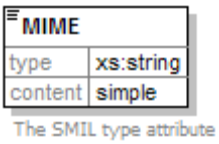


type **xs:string**

annotation
- documentation The SMIL src attribute used in the template

element **infoType/MIME**

diagram

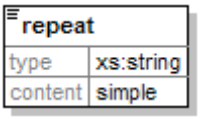


type **xs:string**

annotation documentation The SMIL type attribute

element **infoType/repeat**

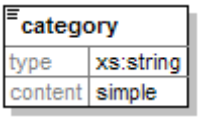
diagram



type **xs:string**

element **infoType/category**

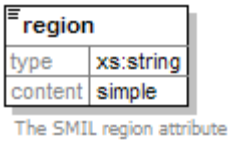
diagram



type **xs:string**

element **infoType/region**

diagram

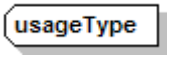


type **xs:string**

annotation documentation The SMIL region attribute

complexType **usageType**

diagram



simpleType **categoryType**

type restriction of **xs:string**

facets
- enumeration slide-show
- enumeration video-clip
- enumeration interactive-music
- enumeration interactive-video
- enumeration hypertext
- enumeration audio-karaoke
- enumeration video-karaoke
- enumeration e-book
- enumeration e-learning
- enumeration tutorial
- enumeration unknown

annotation documentation General categories for the template

simpleType **mediaType**

type restriction of **xs:string**

facets
- enumeration audio
- enumeration img
- enumeration text
- enumeration application
- enumeration video

annotation documentation Media type (i.e.: audio, video, ...)

simpleType **outformatType**

type restriction of **xs:string**

facets
- enumeration SMIL
- enumeration MPEG4
- enumeration HTML

annotation documentation Output formats

simpleType **resourceCategoryType**

type restriction of **xs:string**

facets
- enumeration title
- enumeration subtitle
- enumeration body
- enumeration graphics
- enumeration background
- enumeration multimedia
- enumeration navigation bar
- enumeration control bar
- enumeration footer

annotation documentation General categories for the resource

simpleType **deviceType**

type restriction of **xs:string**

facets
- enumeration PC
- enumeration mobile
- enumeration PDA
- enumeration kiosk

annotation documentation Target device categories

7.6.3 Formal description of style descriptor

The following XML Schema defines the style descriptor (style_descriptor.xsd).

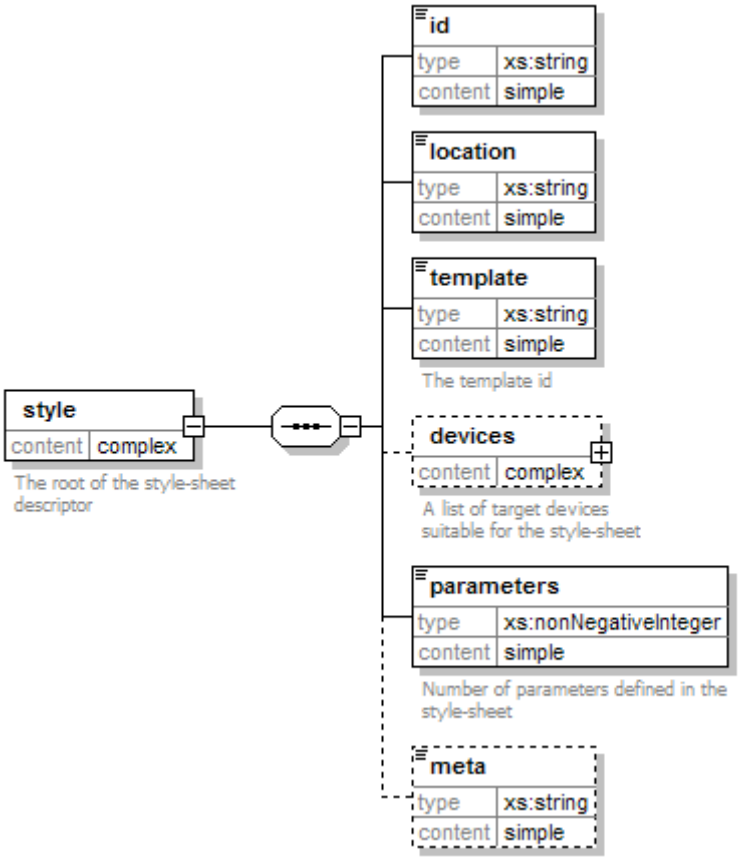
Schema **style_descriptor.xsd**

Elements
style

Simple types
deviceType

element **style**

diagram

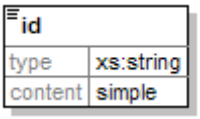


children **id** **location** **template** **devices** **parameters** **meta**

annotation documentation The root of the style-sheet descriptor

element **style/id**

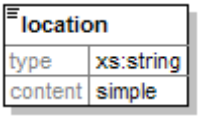
diagram



type **xs:string**

element **style/location**

diagram



type **xs:string**

element **style/template**

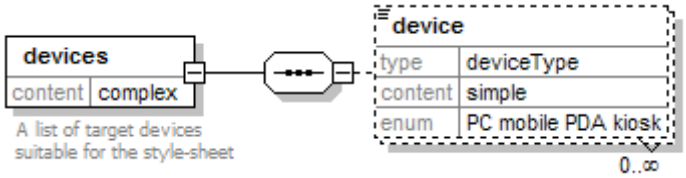
diagram

type **xs:string**

annotation documentation The template id

element **style/devices**

diagram

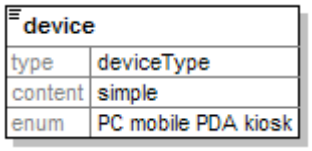


children **device**

annotation documentation A list of target devices suitable for the style-sheet

element **style/devices/device**

diagram

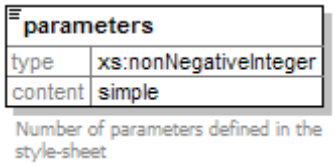


type **deviceType**

facets enumeration PC
enumeration mobile
enumeration PDA
enumeration kiosk

element **style/parameters**

diagram

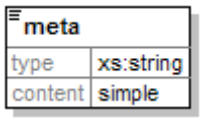


type **xs:nonNegativeInteger**

annotation documentation Number of parameters defined in the style-sheet

element **style/meta**

diagram



type **xs:string**

simpleType **deviceType**

type restriction of **xs:string**

facets enumeration PC
enumeration mobile
enumeration PDA
enumeration kiosk

annotation documentation Target device categories

8 Template Editor and Selector

Tool Profile		
Template Editor and Selector		
Responsible Name	Paolo Vaccari	
Responsible Partner	DSI, EPFL	
Status (proposed/approved)		
Implemented/not implemented	Not implemented	
Status of the implementation		
Executable or Library/module (Support)	Executable	
Single Thread or Multithread	Single Thread	
Language of Development	C++	
Platforms supported	Linux, Mac OS X and MS WIndows	
Reference to the AXFW location of the source code demonstrator		
Reference to the AXFW location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any		
Test cases (present/absent)		
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)	yes	
Usage of the AXMEDIS Error Manager (yes/no)		
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Template descriptor		Template descriptor

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
AXMEDIS SMIL Editor	C++	wxWidgets
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
wxWidgets		

8.1 General Description of the Tool

8.1.1 Template Editor

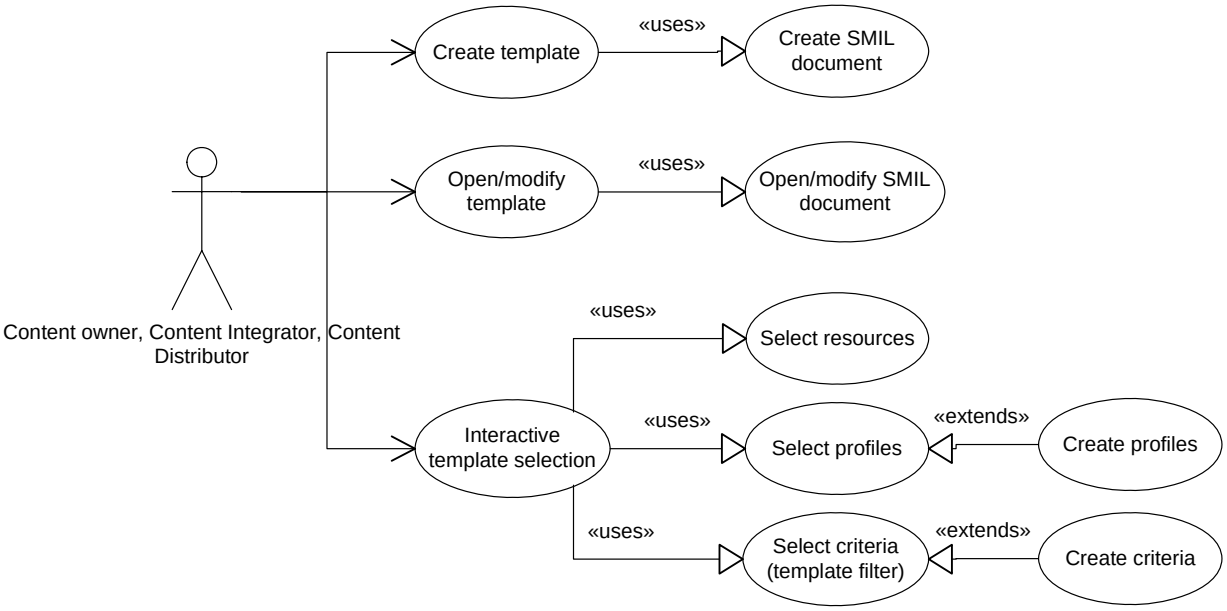
The Template Editor is integrated within the AXMEDIS SMIL Editor and allows user to create new templates or modify existing ones.
The creation of new templates is executed through the creation of a SMIL document: when the new SMIL document is completed, user can save resulting template and style-sheet. Template and style-sheet are obtained from the SMIL document with the application of an XSL Transformation (see formal description of *gentemplate.xsl* and *genstyle.xsl*).
When an existing template is opened, it is displayed with "fake" resources (which may be replaced later with real resources) and it is managed as a normal SMIL document.

8.1.2 Template Selector

The user interface of this Tool also allows an interactive template selection: user selects resources that have to be formatted, and obtains a list of suitable templates. Each template can be previewed and edited through the Template Editor.
The user should provide profiles for user preferences, device capabilities and delivery context: this can be done selecting existing files containing profile descriptors, or creating new descriptors via the user interface.
A criteria indication is also needed: the user can select one of the existing criteria or create a new one via user interace.
After template selection, the interactive formatting process continues through the Style Selector.

8.2 Module Design in terms of Classes

The following diagram describes use-cases of this tool:



8.3 User interface description

Module functionalities are available through the following menu items of the AXMEDIS SMIL editor:

- Menu -> File -> Save template as...
- Menu -> File -> Open template... : *open an existing template, showing "fake" resources*
- Menu -> File -> New criteria... : *allows to create a new set of weights*
- Menu -> Tools -> Template selector... : *allows interactive selection of templates for the given resources (or "fake"resources if real resources have not be selected); the selected template can be previewed as-is, or after the style-sheet application*

8.4 Formal description of format

8.4.1 Formal description of gentemplate.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    This style-sheet processes a complete SMIL document
    and creates an AXMEDIS template.
    -->

<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns:sm="http://www.w3.org/2005/SMIL21/"
            xmlns:ax="http://www.axmedis.org/extensions">
  <xsl:output method="xml" version="1.0" encoding="ISO-8859-1" indent="yes"/>
  <xsl:strip-space elements="*"/>
  <!--xsl:namespace-alias stylesheet-prefix="sm" result-prefix="#default"/-->
  <!-- SMIL structure -->

  <!-- apply templates for smil -->
  <xsl:template match="sm:smil">
    <xsl:comment>
     This template has been automatically
     generated from a SMIL document.
    </xsl:comment>
    <xsl:copy>
      <xsl:if test="@xmlns">
        <xsl:attribute name="xmlns">
          <xsl:value-of select="@xmlns"/>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
```

```
  </xsl:copy>
</xsl:template>

<!-- apply templates for head & body -->
<xsl:template match="sm:head|sm:body">
  <xsl:copy>
    <xsl:attribute name="id">
      <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- attributes for root-layout -->
<xsl:template match="sm:root-layout">
  <xsl:copy>
    <xsl:if test="@title">
      <xsl:attribute name="title">
        <xsl:value-of select="@title"/>
      </xsl:attribute>
    </xsl:if>
  </xsl:copy>
</xsl:template>

<!-- apply templates for layout -->
<xsl:template match="sm:layout">
  <xsl:copy>
    <xsl:attribute name="type">
      <xsl:value-of select="@type"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- attributes for regions -->
<xsl:template match="sm:region">
  <xsl:copy>
    <xsl:attribute name="id">
      <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- copy comments -->
<xsl:template match="comment()">
  <xsl:comment><xsl:value-of select="."/></xsl:comment>
</xsl:template>

<!-- par, seq, excl -->

<!-- apply templates for par, seq and excl -->
<xsl:template match="sm:par|sm:seq|sm:excl">
  <xsl:copy>
    <xsl:if test="@id">
      <xsl:attribute name="id">
        <xsl:value-of select="@id"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@repeatCount">
      <xsl:attribute name="repeatCount">
        <xsl:value-of select="@repeatCount"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@repeatDur">
      <xsl:attribute name="repeatDur">
        <xsl:value-of select="@repeatDur"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@dur">
      <xsl:attribute name="dur">
        <xsl:value-of select="@dur"/>
```

```
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- media -->

<!-- apply templates for img, text, video -->
<xsl:template match="sm:img|sm:text|sm:video">
  <xsl:copy>
    <xsl:attribute name="id">
      <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:attribute name="src">
      <xsl:value-of select="@src"/>
    </xsl:attribute>
    <xsl:if test="@type">
      <xsl:attribute name="type">
        <xsl:value-of select="@type"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:attribute name="region">
      <xsl:value-of select="@region"/>
    </xsl:attribute>
    <xsl:if test="@dur">
      <xsl:attribute name="dur">
        <xsl:value-of select="@dur"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@begin">
      <xsl:attribute name="begin">
        <xsl:value-of select="@begin"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@end">
      <xsl:attribute name="end">
        <xsl:value-of select="@end"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@fit">
      <xsl:attribute name="fit">
        <xsl:value-of select="@fit"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- apply templates for audio and ref -->
<xsl:template match="sm:audio|sm:ref">
  <xsl:copy>
    <xsl:attribute name="id">
      <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:attribute name="src">
      <xsl:value-of select="@src"/>
    </xsl:attribute>
    <xsl:if test="@type">
      <xsl:attribute name="type">
        <xsl:value-of select="@type"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@region">
      <xsl:attribute name="region">
        <xsl:value-of select="@region"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@dur">
      <xsl:attribute name="dur">
        <xsl:value-of select="@dur"/>
      </xsl:attribute>
```

```
    </xsl:if>
    <xsl:if test="@begin">
      <xsl:attribute name="begin">
        <xsl:value-of select="@begin"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@end">
      <xsl:attribute name="end">
        <xsl:value-of select="@end"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@fit">
      <xsl:attribute name="fit">
        <xsl:value-of select="@fit"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- linking elements -->

<!-- apply templates for a and area -->
<xsl:template match="sm:a|sm:area">
  <xsl:copy>
    <xsl:attribute name="href">
      <xsl:value-of select="@href"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

9 Style Editor and Selector

Tool Profile	
Style Editor and Selector	
Responsible Name	Paolo Vaccari
Responsible Partner	DSI, EPFL
Status (proposed/approved)	
Implemented/not implemented	Not implemented
Status of the implementation	
Executable or Library/module (Support)	
Single Thread or Multithread	Single Thread
Language of Development	C++
Platforms supported	Linux, Mac OS X and MS Windows
Reference to the AXFW location of the source code demonstrator	
Reference to the AXFW location of the demonstrator executable tool for internal download	
Reference to the AXFW location of the demonstrator executable tool for public download	
Address for accessing to	

WebServices if any, add accession information (user aNd Passwd) if any		
Test cases (present/absent)		
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)		
Usage of the AXMEDIS Error Manager (yes/no)		
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Style-sheet descriptor		Style-sheet descriptor
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
AXMEDIS SMIL Editor	C++	wxWidgets
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
wxWidgets		

9.1 General Description of the Tool

9.1.1 Style Editor

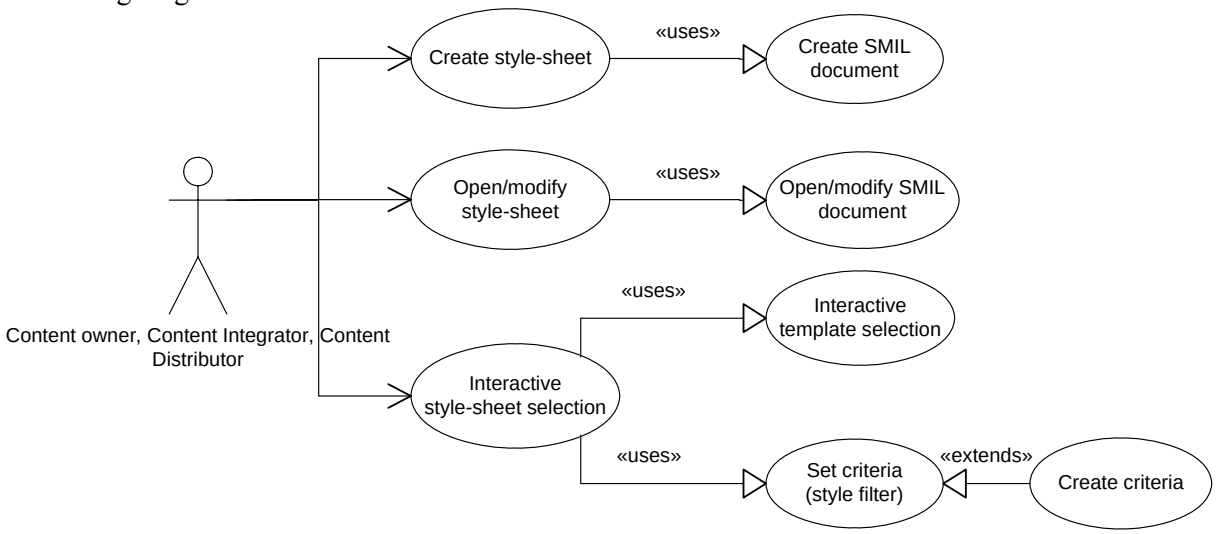
The Style Editor is integrated within the AXMEDIS SMIL Editor and allows user to create new style-sheets or modify existing ones.
The creation of new style-sheets is executed through the creation of a SMIL document: when the new SMIL document is completed, user can save resulting template and style-sheet. Template and style-sheet are obtained from the SMIL document with the application of an XSL Transformation (see formal description of *gentemplate.xsl* and *genstyle.xsl*).
An existing style-sheet can be opened only in association with its related template; the result is displayed with "fake" resources (which may be replaced later with real resources) and it is managed as a normal SMIL document.
In both cases, user may mark some SMIL attributes as "optimizable": the editor creates parameters in the resulting style-sheet and uses such parameters as attribute values.

9.1.2 Style Selector

The user interface of this Tool also allows an interactive style-sheet selection: user first selects resources that have to be formatted, then chooses a template using the Template Selector, finally uses the Style Selector to get a list of style-sheets suitable for the template.
A criteria indication is also needed: the user can select one of the existing criteria or create a new one via user interface.
Each style-sheet can be previewed and edited through the Style Editor, if it doesn't contain parameters. Otherwise, the interactive formatting process must continue with the Style Optimizer.

9.2 Module Design in terms of Classes

The following diagram describes use-cases of this tool:



9.3 User interface description

Module functionalities are available through the following menu items of the AXMEDIS SMIL editor:

- Menu -> File -> Save style-sheet as...
- Menu -> File -> Open style-sheet... : *open an existing style-sheet, if a template has already been selected*
- Menu -> File -> New criteria... : *allows to create a new set of weights*
- Menu -> Tools -> Style-sheet selector... : *allows interactive selection style-sheets for the given template and resources (or "fake"resources if real resources have not be selected)*

9.4 Formal description of format

9.4.1 Formal description of genstyle.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    This style-sheet provides a basic
    system to produce specific style-sheets
    for a SMIL template.
    -->

<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns:ax="http://www.axmedis.org/extensions"
            xmlns:sm="http://www.w3.org/2005/SMIL21/">
  <xsl:output method="xml" version="1.0" encoding="ISO-8859-1" indent="yes"/>
  <xsl:strip-space elements="*"/>
  <!--xsl:namespace-alias stylesheet-prefix="sm" result-prefix="#default"/-->

  <!-- build the basic XSLT structure -->
  <xsl:template match="sm:smil">

    <xsl:comment>
     This style-sheet has been automatically
     generated from a SMIL template
    </xsl:comment>

    <xsl:element name="xsl:stylesheet">
      <xsl:attribute name="version">1.0</xsl:attribute>
      <xsl:attribute name="xmlns:xsl">http://www.w3.org/1999/XSL/Transform</xsl:attribute>
      <xsl:attribute name="xmlns:sm">http://www.w3.org/2005/SMIL21/</xsl:attribute>
      <!-- note: xmlns:sm is only required in xalan, not in sablotron -->
      <xsl:element name="xsl:output">
        <xsl:attribute name="method">xml</xsl:attribute>
        <xsl:attribute name="encoding">ISO-8859-1</xsl:attribute>
        <xsl:attribute name="indent">yes</xsl:attribute>
      </xsl:element>
      <xsl:element name="xsl:strip-space">
        <xsl:attribute name="elements">*</xsl:attribute>
      </xsl:element>
      <!--xsl:element name="xsl:namespace-alias">
          <xsl:attribute name="stylesheet-prefix">sm</xsl:attribute>
          <xsl:attribute name="result-prefix">#default</xsl:attribute>
        </xsl:element-->

       <!--xsl:comment>
           parameters; with sablotron:
           sabcmd xsl input output '$param1=p1' '$param2=p2'
           (their scope includes smilstruct.xsl)
           example:
           xsl:param name="width" select="{@screenWidth}"
         </xsl:comment-->

      <xsl:comment>SMIL structure</xsl:comment>
      <xsl:element name="xsl:include">
        <xsl:attribute name="href">smilstruct.xsl</xsl:attribute>
      </xsl:element>

      <!-- create template for root-layout -->
      <xsl:comment>attributes for root-layout</xsl:comment>
      <xsl:element name="xsl:template">
        <xsl:attribute name="match">sm:root-layout</xsl:attribute>
        <xsl:comment>create root-layout element</xsl:comment>
        <xsl:element name="xsl:copy">
          <xsl:comment>copy root-layout attributes</xsl:comment>
          <xsl:for-each select="//sm:root-layout/@*">
            <xsl:element name="xsl:attribute">
              <xsl:attribute name="name"><xsl:value-of select="name()"/></xsl:attribute>
              <xsl:value-of select="."/>
            </xsl:element>
```

```
        </xsl:for-each>
      </xsl:element> <!-- copy -->
    </xsl:element> <!-- root-layout -->

    <!-- create template for regions -->
    <xsl:comment>attributes for regions</xsl:comment>
    <xsl:element name="xsl:template">
      <xsl:attribute name="match">sm:region</xsl:attribute>
      <xsl:element name="xsl:choose">
        <xsl:apply-templates select=".//sm:region | .//ax:loop"/>
        <xsl:element name="xsl:otherwise">
          <xsl:comment>include the region as is</xsl:comment>
          <xsl:element name="xsl:copy-of">
            <xsl:attribute name="select">.</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:element>

  </xsl:element><!-- xsl:stylesheet -->
</xsl:template>

<xsl:template match="sm:region">
  <xsl:element name="xsl:when">
    <xsl:attribute name="test">@id=&apos;<xsl:value-of select="@id"/>&apos;</xsl:attribute>
    <xsl:comment>copy element</xsl:comment>
    <xsl:element name="xsl:copy">
      <xsl:comment>copy attributes</xsl:comment>
      <xsl:for-each select="@*">
        <xsl:element name="xsl:attribute">
          <xsl:attribute name="name"><xsl:value-of select="name()"/></xsl:attribute>
          <xsl:value-of select="."/>
        </xsl:element>
      </xsl:for-each>
      <xsl:element name="xsl:apply-templates"/>
    </xsl:element> <!-- copy -->
  </xsl:element> <!-- when -->
</xsl:template> <!-- region -->

</xsl:stylesheet>
```

9.4.2 Formal description of smilstruct.xsl

The *smilstruct.xsl* file has to be located in the same directory of each style-sheet produced by the formatting system.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    This style-sheet processes a SMIL document
    and reproduces its basic structure.
    To be included into a more specific style-sheet.
    -->

<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns:ax="http://www.axmedis.org/extensions"
            xmlns:sm="http://www.w3.org/2005/SMIL21/">
  <xsl:output method="xml" encoding="ISO-8859-1" indent="yes"/>
  <xsl:strip-space elements="*"/>
  <!--xsl:namespace-alias stylesheet-prefix="sm" result-prefix="#default"/-->

  <!-- SMIL structure -->

  <!-- apply templates for smil, head & body -->
  <xsl:template match="sm:smil|sm:head|sm:body">
    <xsl:copy>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>
```

```
<!-- apply templates for layout -->
<xsl:template match="sm:layout">
  <xsl:copy>
    <xsl:attribute name="type">
      <xsl:value-of select="@type"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- copy comments -->
<xsl:template match="comment()">
  <xsl:comment><xsl:value-of select="."/></xsl:comment>
</xsl:template>

<!-- apply templates for par, seq and excl -->
<xsl:template match="sm:par|sm:seq|sm:excl">
  <!-- propagate $index for ax:repeat
       <xsl:param name="index" select="1"/-->
  <!-- copy element -->
  <xsl:copy>
    <xsl:for-each select="@*">
      <!-- copy attributes -->
      <xsl:copy/>
    </xsl:for-each>

    <!--xsl:if test="@id">
      <xsl:attribute name="id">
        <xsl:value-of select="@id"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@repeatCount">
      <xsl:attribute name="repeatCount">
        <xsl:value-of select="@repeatCount"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@repeatDur">
      <xsl:attribute name="repeatDur">
        <xsl:value-of select="@repeatDur"/>
      </xsl:attribute>
    </xsl:if-->
    <xsl:apply-templates>
      <!--xsl:with-param name="index" select="$index"/-->
    </xsl:apply-templates>
  </xsl:copy>
</xsl:template>

<!-- media -->

<!-- apply templates for img, text, video, audio and ref -->
<xsl:template match="sm:img|sm:text|sm:video|sm:audio|sm:ref">
  <!-- copy element -->
  <xsl:copy>
    <xsl:for-each select="@*">
      <!-- copy attributes -->
      <xsl:copy/>
    </xsl:for-each>
    <!--xsl:attribute name="id">
      <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:attribute name="src">
      <xsl:value-of select="@src"/>
    </xsl:attribute>
    <xsl:attribute name="type">
      <xsl:value-of select="@type"/>
    </xsl:attribute>
    <xsl:attribute name="region">
      <xsl:value-of select="@region"/>
    </xsl:attribute>
    <xsl:if test="@dur">
      <xsl:attribute name="dur">
        <xsl:value-of select="@dur"/>
```

```
      </xsl:attribute>
    </xsl:if-->
    <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>

 <!-- linking elements -->

 <!-- apply templates for a and area -->
 <xsl:template match="sm:a|sm:area">
  <xsl:copy>
   <xsl:attribute name="href">
    <xsl:value-of select="@href"/>
   </xsl:attribute>
   <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>

 <!-- following section is EXPERIMENTAL -->

 <!-- switch -->

 <!-- apply templates for switch, using my preferences -->
 <xsl:template match="sm:switch">
  <!-- select all nodes that satisfy at least one condition
      (which maybe is not the behaviour we want)
      -->
  <xsl:apply-templates
select="*[@systemLanguage='en']|*[@systemScreenDepth='8']|*[@systemCaptions='on']|*[@systemOperatingSyst
em='linux']|*[@systemScreenSize='800x600']"/>
  <!-- select all nodes that satisfy all conditions
      (which is not the behaviour we want, for sure!)
      <xsl:apply-templates select="*[@systemLanguage='en']
                          [@systemScreenDepth='16']
                          [@systemCaptions='on']
                          [@systemOperatingSystem='linux']
                          [@systemScreenSize='800x600']"/>
      -->
    </xsl:template>

    <!--
       <xsl:choose>
         <xsl:when test="$mode='en'">
          <xsl:apply-templates select="*[@systemLanguage='en']"/>
         </xsl:when>
         <xsl:when test="$mode='it'">
          <xsl:apply-templates select="*[@systemLanguage='it']"/>
         </xsl:when>
       </xsl:choose>
       -->

 <!-- AXMEDIS extensions to SMIL -->

 <!-- apply templates for ax:oldrepeat -->
 <xsl:template match="ax:oldrepeat">
  <!-- the item to be processed -->
  <xsl:param name="index" select="1"/>
  <!-- process ax:items -->
  <xsl:for-each select="descendant::ax:items"><!-- only 1 level of repeat! -->
   <xsl:element name="{@element}">
    <xsl:copy-of select="@*[not(name()='element')]"/>
    <xsl:copy-of select="ax:item[$index]/@*"/>
   </xsl:element>
  </xsl:for-each>
  <!-- check if other items have to be processed -->
  <xsl:if test="$index &lt; count(descendant-or-self::ax:items[1]/ax:item)">
   <!-- continue recursively -->
   <xsl:apply-templates select="self::ax:repeat">
    <xsl:with-param name="index" select="$index + 1"/>
   </xsl:apply-templates>
  </xsl:if>
 </xsl:template>
```

```
  <!-- apply templates for ax:repeat -->
  <xsl:template match="ax:repeat">
    <!-- the item to be processed -->
    <xsl:param name="index" select="1"/>
    <!-- process items and other elements (par, seq & excl) -->
    <!-- rem: par, seq & excl have to propagate $index -->
    <xsl:apply-templates>
      <xsl:with-param name="index" select="$index"/>
    </xsl:apply-templates>
    <!-- check if other items have to be processed -->
    <xsl:if test="$index &lt; count(descendant-or-self::ax:items[1]/ax:item)">
      <!-- continue recursively -->
      <xsl:apply-templates select="self::ax:repeat">
        <xsl:with-param name="index" select="$index + 1"/>
      </xsl:apply-templates>
    </xsl:if>
  </xsl:template>

  <!-- apply templates for ax:items -->
  <xsl:template match="ax:items">
    <xsl:param name="index"/>
    <xsl:element name="{@element}">
      <xsl:copy-of select="@*[not(name()='element')]"/>
      <xsl:copy-of select="ax:item[$index]/@*"/>
    </xsl:element>
  </xsl:template>

</xsl:stylesheet>
```

10 Style Optimizer

Tool Profile	
Style Optimizer	
Responsible Name	Paolo Vaccari
Responsible Partner	DSI
Status (proposed/approved)	
Implemented/not implemented	Not implemented
Status of the implementation	
Executable or Library/module (Support)	Executable
Single Thread or Multithread	Single Thread
Language of Development	C++
Platforms supported	Linux, Mac OS X and MS Windows
Reference to the AXFW location of the source code demonstrator	
Reference to the AXFW location of the demonstrator executable tool for internal download	
Reference to the AXFW location of the demonstrator executable tool for public download	
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any	

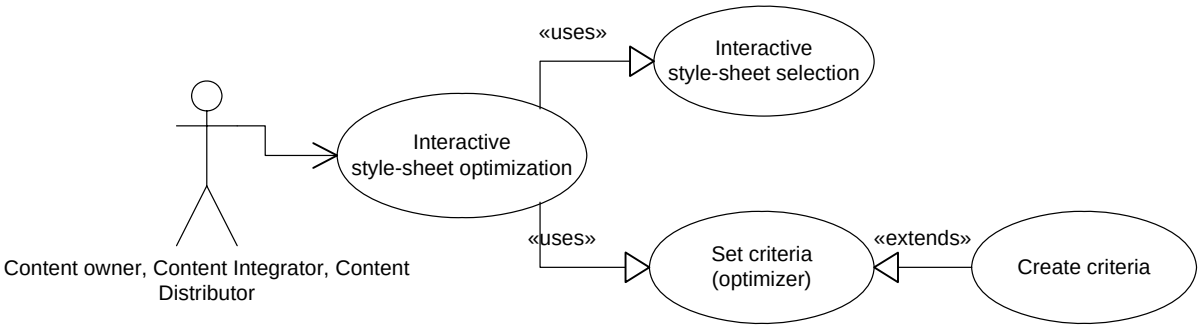
Test cases (present/absent)		
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)		
Usage of the AXMEDIS Error Manager (yes/no)		
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
AXMEDIS SMIL Editor	C++	wxWidgets
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
wxWidgets		

10.1 General Description of the Module

The Style Optimizer is integrated within the AXMEDIS SMIL Editor and allows user to get optimized value for parameters defined in the selected style-sheet. User may accept proposed values or set custom values.

10.2 Module Design in terms of Classes

The following diagram describes use-cases of this tool:



10.3 User interface description

Module functionalities are available through the following menu items of the AXMEDIS SMIL editor:

- Menu -> File -> New criteria... : *allows to create a new set of weights*
- Menu -> Tools -> Optimizer... : *allows optimization of the selected style-sheets; optimized values are proposed to the user, who may change them and use custom values*

11 AXMEDIS DATA Types and Functions for JavaScript

Module/Tool Profile		
AXMEDIS DATA Types and Functions for JavaScript		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)	approved	
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	MS Windows, Linux	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any	Specified in the corresponding JS Class table	
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	Yes	
Usage of the AXMEDIS Error Manager (yes/no)	NO	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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
JS API Spidermonkey	JS API v.1.5	LGPL Licence

11.1 General Description of the Module

In addition to using the engine's built-in objects, it is possible to create, initialize, and use own JS objects and JS Functions. This is especially true if the JS engine is used with scripts to automate the application. Custom JS objects can provide direct program services, or they can serve as interfaces to program's services. For example, a custom JS object that provides direct service might be one that handles all of an application's network access, or might serve as an intermediary broker of database services. Or a JS object that mirrors data and functions that already exist in the application may provide an object-oriented interface to C code that is not otherwise, strictly-speaking, object-oriented itself. Such a custom object acts as an interface to the application itself, passing values from the application to the user, and receiving and processing user input before returning it to the application. Such an object might also be used to provide access control to the underlying functions of the application.
There are two ways to create custom objects that the JS engine can use:

- Write a JS script that creates an object, its properties, methods, and constructor, and then pass the script to the JS engine at run time.
- Embed code (wrapping) in the application that defines the object's properties and methods, call the engine to initialize a new object, and then set the object's properties through additional engine calls. An advantage of this method is that the application can contain native methods that directly manipulate the object embedding.

11.2 A JavaScript class and functions in C++

In this section is reported an example of how to wrap a C++ class by embedding code without using inheritance ([2], [3]). For more details about JS API please see the official site of SpiderMonkey on Mozilla web page. The class used in the example is the following:

```
class Customer
{
public:
 int GetAge() { return m_age; }
 void SetAge(int newAge) { m_age = newAge; }
 std::string GetName() { return m_name; }
 void SetName(std::string newName) { m_name = newName; }

private:
 int m_age;
 std::string m_name;
};
```

11.2.1 Step 1 - The JavaScript class.

Create a new C++ class that derives from the C++ class you want to use in JavaScript or create a new C++ class which has a member of the type of that C++ class.
A class is defined in JavaScript with a JSClass structure. Create a static member of this type. Declare it as a public member, because this structure can be useful for other classes. It can be used by other classes to determine the type of an object. (see JS_InstanceOf API)

```
// JSCustomer.h
class JSCustomer
{
 public:
  JSCustomer() : m_pCustomer(NULL)
  {
  }

  ~JSCustomer()
  {
    delete m_pCustomer;
    m_pCustomer = NULL;
  }

  static JSClass customerClass;

 protected:
  void setCustomer(Customer *customer)
  {
    m_pCustomer = customer;
  }

  Customer* getCustomer()
  {
    return m_pCustomer;
  }

 private:
  Customer *m_pCustomer;

};
```

The JSClass structure contains the name of the JavaScript class, some flags and the name of callbacks used by the engine. For example a callback is used when the engine needs to retrieve a property from your class. Define the JSClass structure in the implementation file of the C++ class as below.

```
// JSCustomer.cpp
JSClass JSCustomer::customerClass =
{
 "Customer", JSCLASS_HAS_PRIVATE,
 JS_PropertyStub, JS_PropertyStub,
 JSCustomer::JSGetProperty, JSCustomer::JSSetProperty,
 JS_EnumerateStub, JS_ResolveStub,
 JS_ConvertStub, JSCustomer::JSDestructor
```

```
};
```

The used callbacks are JSCustomer::JSGetProperty, JSCustomer::JSSetProperty and JSCustomer::JSDestructor. JSGetProperty is called when the engine needs a property, JSSetProperty is called when the engine sets a property and JSDestructor is called when the JavaScript object is destroyed.
The flag JSCLASS_HAS_PRIVATE is used so that the engine provides memory you can use to attach some data to a JavaScript object. You can use this to store a pointer to your class.
The callbacks are static member functions of the C++ class.

```
static JSBool JSGetProperty(JSContext *cx, JSObject *obj, jsval id, jsval *vp);
static JSBool JSSetProperty(JSContext *cx, JSObject *obj, jsval id, jsval *vp);
static JSBool JSConstructor(JSContext *cx, JSObject *obj, uintN argc,
                            jsval *argv, jsval *rval);
static void JSDestructor(JSContext *cx, JSObject *obj);
```

11.2.2 Step 2 - Initialize your JavaScript object

Create another static method called JSInit. See below for an example. This method will be called by the Application that creates the JavaScript runtime.

```
static JSObject *JSInit(JSContext *cx, JSObject *obj, JSObject *proto);
```

The implementation looks like this

```
JSObject *JSCustomer::JSInit(JSContext *cx, JSObject *obj, JSObject *proto)
{
  JSObject *newObj = JS_InitClass(cx, obj, proto, &customerClass,
                                  JSCustomer::JSConstructor, 0,
                                  NULL, JSCustomer::customer_methods,
                                  NULL, NULL);
  JS_DefineProperties(cx, newObj, JSCustomer::customer_properties);
  return newObj;
}
```

The static method JSConstructor will be called when your object is instantiated in a script. This method is very handy to attach your data to the object using the JS_SetPrivate API.

```
 JSBool JSCustomer::JSConstructor(JSContext *cx, JSObject *obj, uintN argc,
                                  jsval *argv, jsval *rval)
 {
   JSCustomer *p = new JSCustomer();
   p->setCustomer(new Customer());
   JS_SetPrivate(cx, obj, p);
   return JS_TRUE;
 }
```

This constructor method can have multiple arguments, which you can use to initialize your class. Now that you've created a pointer on the heap, you also need a way to destroy the pointer. This is done in the static method JS_Destructor.

```
void JSCustomer::JSDestructor(JSContext *cx, JSObject *obj)
{
  JSCustomer *p = JS_GetPrivate(cx, obj);
  delete p;
  p = NULL;
}
```

11.2.3 Step 3 - Adding properties

Add a static array member of the type JSPropertySpec. This array will contain the information of a property. Create also an enum for the property ids.

```
static JSPropertySpec customer_properties[];
enum
{
 name_prop,
 age_prop
};
```

Initialize this array in the implementation file as follows

```
JSPropertySpec JSCustomer::customer_properties[] =
{
  { "name", name_prop, JSPROP_ENUMERATE },
  { "age", age_prop, JSPROP_ENUMERATE },
  { 0 }
};
```

The last element of the array must be a null element. Each element contains another array with 3 elements. The first element is the name that will be used in JavaScript. The second is a unique id for the property. This will be passed to the callback functions. And the third one is a flag. JSPROP_ENUMERATE means that a script will see this property when it's enumerating the properties of the Customer object. You can also specify JSPROP_READONLY to indicate that the property can't be changed in the script.

Now you can implement the callbacks for getting and setting properties.

```
JSBool JSCustomer::JSGetProperty(JSContext *cx, JSObject *obj, jsval id, jsval *vp)
{
  if (JSVAL_IS_INT(id))
  {
    Customer *priv = (Customer *) JS_GetPrivate(cx, obj);
    switch(JSVAL_TO_INT(id))
    {
      case name_prop:

        break;
      case age_prop:
        *vp = INT_TO_JSVAL(priv->getCustomer()->GetAge());
        break;
    }
  }
  return JS_TRUE;
}

JSBool JSCustomer::JSSetProperty(JSContext *cx, JSObject *obj, jsval id, jsval *vp)
{
  if (JSVAL_IS_INT(id))
  {
    Customer *priv = (Customer *) JS_GetPrivate(cx, obj);
    switch(JSVAL_TO_INT(id))
    {
      case name_prop:
        break;
      case age_prop:
        priv->getCustomer()->SetAge(JSVAL_TO_INT(*vp));
        break;
    }
  }
  return JS_TRUE;
}
```

It's recommanded to return JS_TRUE in the property callbacks. When you return JS_FALSE a prototype will not be searched when the property is not found in your object.

11.2.4 Step 4 - Adding methods

Create a static member array of JSFunctionSpec type.

```
static JSFunctionSpec customer_methods[];
```

Initialize this array in the implementation file as follows

```
JSFunctionSpec wxJSFrame::wxFrame_methods[] =
{
  { "computeReduction", computeReduction, 1, 0, 0 },
  { 0 }
};
```

The last element of the array must always be a null element. Each element is another array with 5 elements. The first element is the name of the method that's used in the script. The second one is the name of a global or static member function. The third element is the number of arguments of this method. The last two elements are ignored.

Create a static method in the class

```
 static JSBool computeReduction(JSContext *cx, JSObject *obj, uintN argc,
                                jsval *argv, jsval *rval);
```

You return JS_TRUE when the function is successful. Otherwise you return JS_FALSE. The actual return of your JavaScript method is placed in the rval argument.

A sample implementation of this method

```
JSBool JSCustomer::computeReduction(JSContext *cx, JSObject *obj, uintN argc,
                                    jsval *argv, jsval *rval)
{
  JSCustomer *p = JS_GetPrivate(cx, obj);
  if ( p->getCustomer()->GetAge() < 25 )
    *rval = INT_TO_JSVAL(10);
  else
    *rval = INT_TO_JSVAL(5);
  return JS_TRUE;
}
```

11.2.5 An example

The following script uses the previously created object

```
var c = new Customer();
c.name = "Franky";
c.age = 32;
var reduction = c.computeReduction();
```

Don't forget to initialize the JavaScript object when you create the context:

```
JSObject *obj = JSCustomer::JSInit(cx, global);
```

11.2.6 Wrapping functions

To wrap a native function you use JS_DefineFunction or to register multiple functions with one API call JS_DefineFunctions.

```
/* Define a bunch of native functions first: */
static JSBool
my_abs(JSContext *cx, JSObject *obj, uintN argc, jsval *argv, jsval *rval)
{
  jsdouble x, z;

  if (!JS_ValueToNumber(cx, argv[0], &x))
    return JS_FALSE;
  z = (x < 0) ? -x : x;
  return JS_NewDoubleValue(cx, z, rval);
}

. . .

/*
 * Use a JSFunctionSpec array terminated with a null name to define a
 * bunch of native functions.
 */
static JSFunctionSpec my_functions[] = {
  /*    name          native          nargs    */
  {"abs",         my_abs,         1},
  {"acos",        my_acos,        1},
  {"asin",        my_asin,        1},
  . . .
  {0}
};

/*
 * Pass a particular object to define methods for it alone.  If you pass
 * a prototype object, the methods will apply to all instances past and
 * future of the prototype's class (see below for classes).
 */
JS_DefineFunctions(cx, globalObj, my_functions);
```

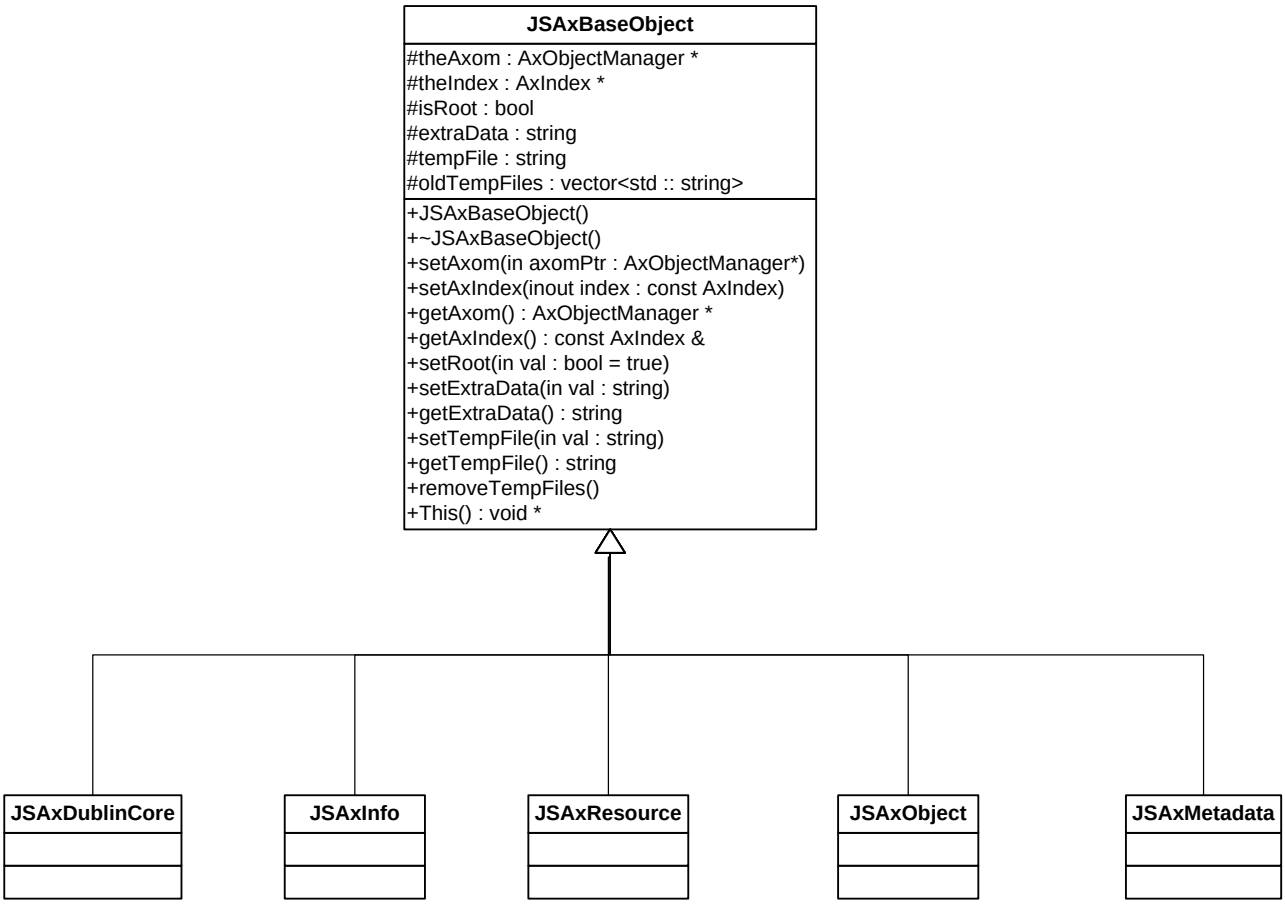
11.3 JSAXOM: AXMEDIS Data Model JS wrapping (DSI)

The JSAXOM models a set of JS Classes that wrap the main classes of the AXMEDIS Object Model. It includes the:

- JSAxObject for managing an instance of the AXMEDIS Object
- JSAxInfo for managing the metadata of the AXInfo section.
- JSAxResource for managing digital resources embedded or to be embedded in the AXMEDIS Object.
- JSAxDublinCore for managing an instance of Dublin Core metadata section
- JSAxMetadata for managing an instance of an AxMetadata object to cope with generic metadata

11.3.1 Module Design in terms of Classes

In this section the Axmedis Data Model wrapping for the Javascript and the current class diagram are reported.



UML Class Diagram of JSAxClasses for the AXMEDIS Data Model in Javascript

JSAxBaseObject Class – This class is the base class for *JSAxObject*, *JSAxDublinCore, JSAxMetadata, JSAxInfo* and *JSAxResource* classes. It allows linking them to the current instance of the Axmedis Object Manager (*theAxom*) and accessing to its methods and commands. It allows also to associate a reference to *theIndex* (*AxIndex*) with the current instance of a child class when it is inserted into the Axmedis Object Data Model. The *isRoot* attribute is set at TRUE when the instance refers to the root object of the Axmedis Object Data Model.

JSAxBaseObject
#theAxom : AxObjectManager * #theIndex : AxIndex * #isRoot : bool #extraData : string #tempFile : string #oldTempFiles : vector<std :: string>
+JSAxBaseObject() +~JSAxBaseObject() +setAxom(in axomPtr : AxObjectManager*) +setAxIndex(inout index : const AxIndex) +getAxom() : AxObjectManager * +getAxIndex() : const AxIndex & +setRoot(in val : bool = true) +setExtraData(in val : string) +getExtraData() : string +setTempFile(in val : string) +getTempFile() : string +removeTempFiles() +This() : void *

JSAxResource Class – The class models the AxResource type of the Axmedis Framework, it is used to store digital resource (audio, video, image, text, etc…) both raw data and Axmedis digital component.
This class wraps the AxResource class and provides functionalities exposing setter and getter methods to:

- access to the mime type
- access to the byte stream of the resource
- create a new resource and to embed a file or a reference inside a resource object

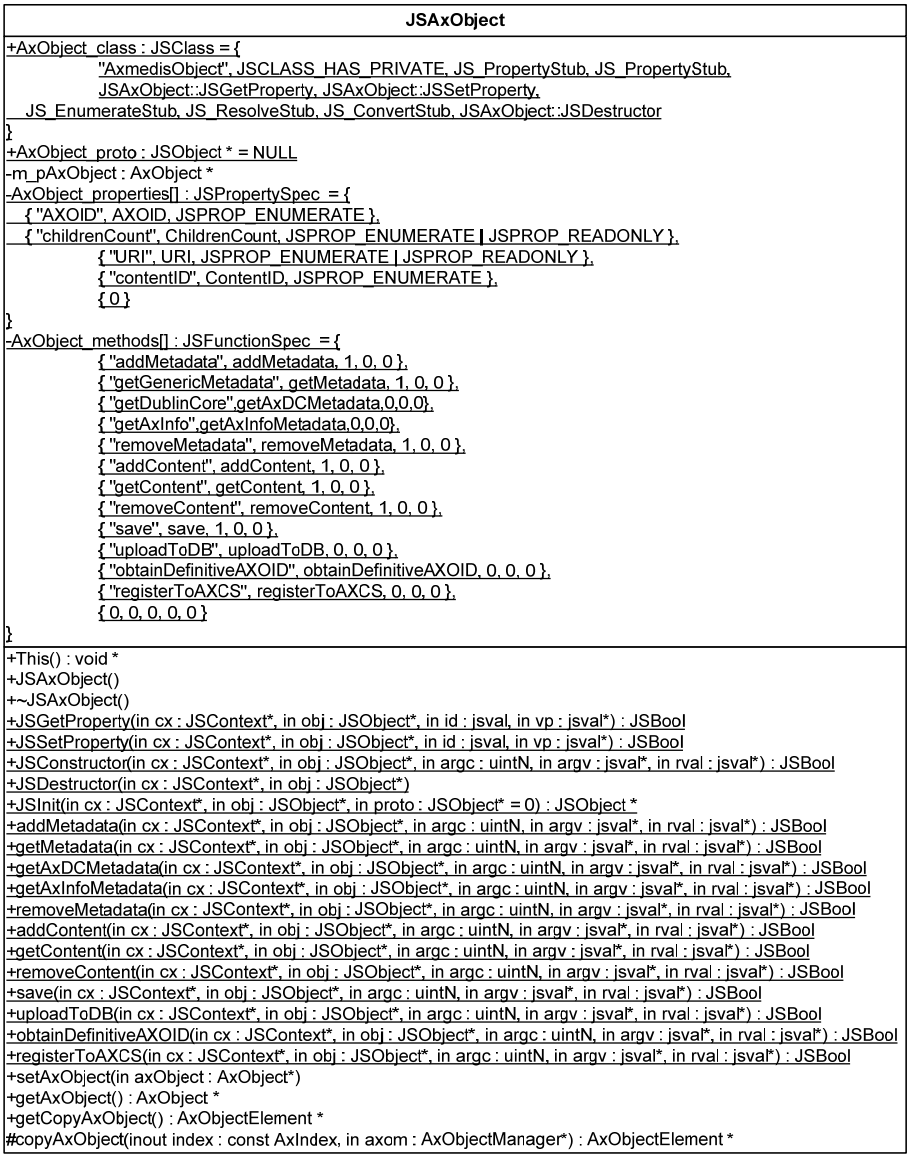
JSAxResource

+AxResource_class : JSClass = {
"AxResource", JSCLASS_HAS_PRIVATE, JS_PropertyStub, JS_PropertyStub,
JSAxResource::JSGetProperty, JSAxResource::JSSetProperty,
JS_EnumerateStub, JS_ResolveStub, JS_ConvertStub, JSAxResource::JSDestructor
}
+AxResource_proto : JSObject * = NULL
-m_pAxResource : AxResource *
-AxResource_properties[] : JSPropertySpec = {
{ "mimeType", MimeType, JSPROP_ENUMERATE },
{ "ref", Ref, JSPROP_ENUMERATE },
{ "contentID",ContentID, JSPROP_ENUMERATE },
{ "encoding",Encoding, JSPROP_ENUMERATE },
{ 0 }
}
-AxResource_methods[] : JSFunctionSpec = {
{ "load", loadResource, 1, 0, 0 },
{ "save", saveResource, 1, 0, 0 },
//{ "clear", clearResource, 0, 0, 0 },
{ 0, 0, 0, 0, 0 }
}

+This() : void * +JSAxResource() +~JSAxResource() +JSGetProperty(in cx : JSContext*, in obj : JSObject*, in id : jsval, in vp : jsval*) : JSBool +JSSetProperty(in cx : JSContext*, in obj : JSObject*, in id : jsval, in vp : jsval*) : JSBool +JSConstructor(in cx : JSContext*, in obj : JSObject*, in argc : uintN, in argv : jsval*, in rval : jsval*) : JSBool +JSDestructor(in cx : JSContext*, in obj : JSObject*) +JSInit(in cx : JSContext*, in obj : JSObject*, in proto : JSObject* = 0) : JSObject * +loadResource(in cx : JSContext*, in obj : JSObject*, in argc : uintN, in argv : jsval*, in rval : jsval*) : JSBool +clearResource(in cx : JSContext*, in obj : JSObject*, in argc : uintN, in argv : jsval*, in rval : jsval*) : JSBool +saveResource(in cx : JSContext*, in obj : JSObject*, in argc : uintN, in argv : jsval*, in rval : jsval*) : JSBool +setAxResource(in AxResource : AxResource*) +getAxResource() : AxResource * #findMimeType(in ext : string) : string

JSAxObject Class – It is the mapping of an AXMEDIS Object for JavaScript. According to the specification of the AXMEDIS OBJECT MANAGER and Axmedis Data Model, it provides and wraps methods to:

- Create an empty AXMEDIS object with own AXOM by instantiating a new Axmedis Object.
- Create and fill an AXMEDIS object with own AXOM by loading content from an URL.
- Add/Remove an Element/Object to the AXMEDIS object. The addition of an element returns the new object reference inside the Axmedis Object.
- Get all Elements/Objects. It returns a Javascript array of Element/Objects.
- Add Resource, it adds a digital resource (audio, video, text, etc…) to a specific Element/Object. It returns the new resource object reference inside the Axmedis Object.
- Remove a Resource (audio, video, text, etc…) by using the object Resource reference.
- Get Resources. It returns a Javascript array of Resource objects.
- Add an AXInfo, Dublin Core or generic metadata object. It returns the new metadata object reference inside the Axmedis Object.
- Get the AXInfo, the Dublin Core metadata
- Remove any metadata object.

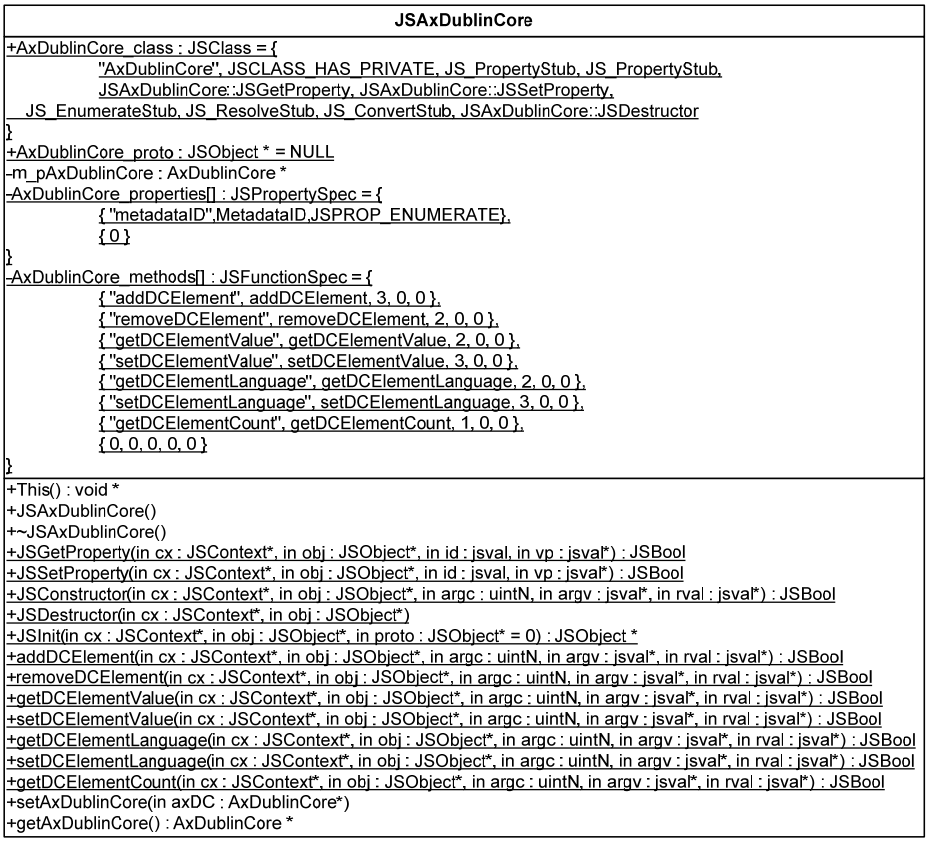


JSAxInfo Class – It maps and allows managing the metadata of the AXINFO in the JavaScript. This class manages the access to individual elements and fields in AXINFO metadata, this class map all the functionalities provided by AxInfo class exposing setter and getter methods for accessing to data. It allows to manage:

- ObjectCreator information
- Owner information
- Distributor information
- Object Status information
- PromoOf information
- Workflow information
- Fingerprints information
- PAR information

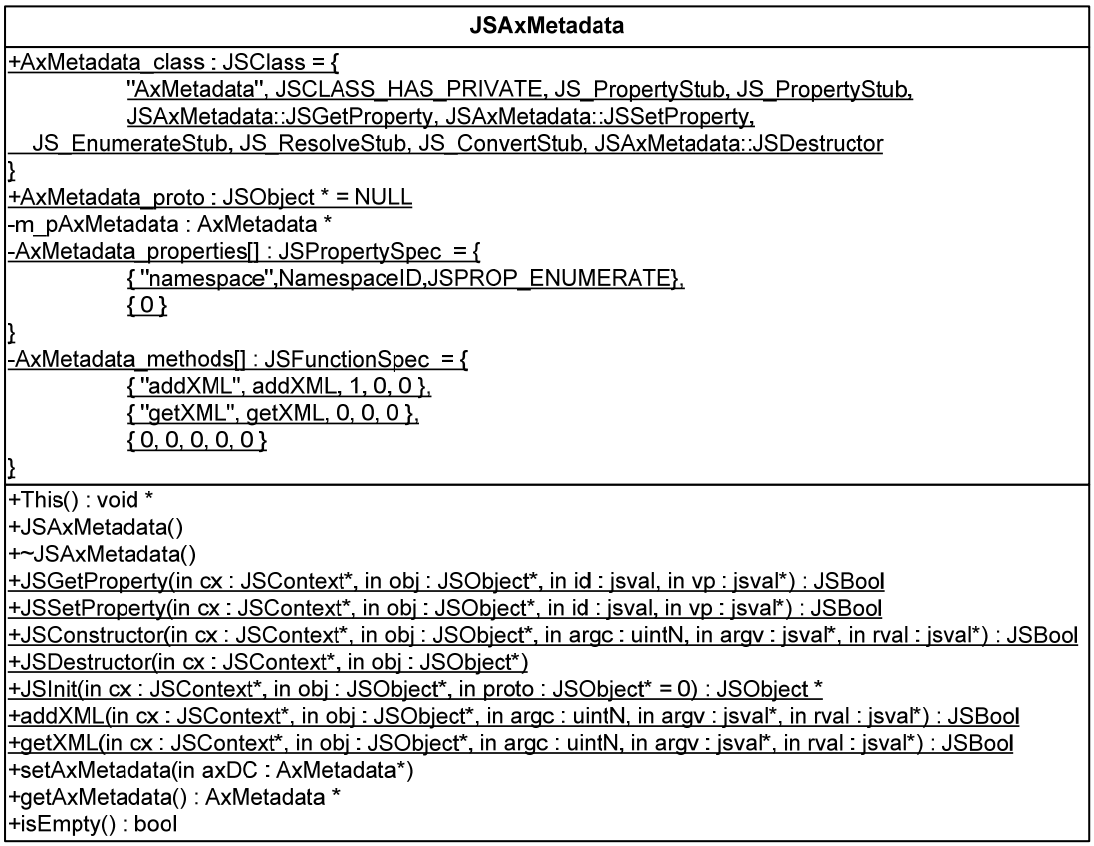
JSAxDublinCore Class - The class models the AxDublinCore type of the Axmedis Framework, it is used to store metadata in Dublin Core format.
This class d provides functionalities exposing setter and getter methods to:

- retrieve the description of metadata
- to store the metatada
- to get the number of elements
- to remove an element



JSAxMetadata Class - The class models the AxMetadata type of the Axmedis Framework, it is used to store metadata in XML format.
This class d provides functionalities exposing setter and getter methods to:

- retrieve the XML description of metadata
- to store the XML of metatada



11.3.2 Draft User Manual

In this section a draft description of the JSAXOM classes is provided in terms of Javascript language. For each class the set of exposed attributes, constructors and methods are reported.

JSAxObject

The Class name in the Javascript is *AxmedisObject*

It is the mapping of an AXMEDIS Object in JavaScript. According to the specification of the AXMEDIS OBJECT MANAGER and Axmedis Data Model, it provides and wraps methods to:

• Create an empty AXMEDIS object with own AXOM by instantiating a new Axmedis Object.
• Create and fill an AXMEDIS object with own AXOM by loading content from an URL.
• Add/Remove an Element/Object to the AXMEDIS object. The addition of an element returns the new object reference inside the Axmedis Object.
• Get all Elements/Objects. It returns a Javascript array of Element/Objects.
• Add Resource, it adds a digital resource (audio, video, text, etc…) to a specific Element/Object. It returns the new resource object reference inside the Axmedis Object.
• Remove a Resource (audio, video, text, etc…) by using the object Resource reference.
• Get Resources. It returns a Javascript array of Resource objects.
• Add an AXInfo, Dublin Core or generic metadata object. It returns the new metadata object reference inside the Axmedis Object.
• Get the AXInfo, the Dublin Core metadata
• Remove any metadata object.

Exposed properties:

string AXOID
It provides the current axoid

number childrenCount
It represemts the number of Children items

string URI
Where the object is located

string contentID
The content identifier

Exposed methods:

AxmedisObject()
Empty constructor for an empty AXMEDIS Object.

AxmedisObject (string URI)
Constructor with an URI parameter. The Axmedis Object is loaded by using the URI. It can specifies or a file system or a database location.

AxmedisObject (string AXOID, int version, boolean lock, string AXDBLoadEndPoint,string AXDBUser, string AXDBPasswd, string AXDBLockUnloadEndpoint)
Constructor with AXMEDIS Database parameters. The Axmedis Object is loaded by using the *AXOID*, the *version*. The *lock* flag when *true* allows locking the object. The *version* set to -1 allows retrieving the last version of the object.

setAXOID(string axoid)
It sets the axoid of the object.

string getAXOID()
It returns the current AXOID.

addMetadata(AxInfo obj)
It adds an AxInfo metadata objects AxInfo.

addMetadata(AxDublinCore obj)
It adds an AxDublinCore metadata.

addMetadata(AxMetadata obj)
It adds an AxMetadata object.

array getGenericMetadata()
It returns an array of generic metadata objects or null.

AxDublinCore getDublinCore()
It returns an AxDublinCore metadata object.

AxInfo getAxInfo()
It returns an AxInfo metadata object.

removeMetadata (AxInfo obj)
It removes an AxInfo metadata objects AxInfo.

removeMetadata (AxDublinCore obj)
It removes an AxDublinCore metadata.

removeMetadata (AxMetadata obj)
It removes a generic metadata object.

addContent(AxResource res)
It adds an AxResource content instance

addContent(AxmedisObject res)
It adds an AxmedisObject content instance

addContent(Document res)
It adds a Document coming from Searchbox tool transforming it in AxResource

insertContent(AxResource res1, AxResource res2, boolean insertBefore)
It inserts the *res1* before/after *res2* in the root AxmedisObject. The *insertBefore* flags allows inserting before (*true*) or after

insertContent(AxmedisObject res1, AxResource res2, boolean insertBefore)
It inserts the *res1* before/after *res2* in the root AxmedisObject. The *insertBefore* flags allows inserting before (*true*) or after

insertContent(AxmedisObject res1, AxmedisObject res2, boolean insertBefore)
It inserts the *res1* before/after *res2* in the root AxmedisObject. The *insertBefore* flags allows inserting before (*true*) or after

insertContent(AxResource res1, AxmedisObject res2, boolean insertBefore)
It inserts the *res1* before/after *res2* in the root AxmedisObject. The *insertBefore* flags allows inserting before (*true*) or after

boolean moveContent(AxResource res1, AxmedisObject parentOfRes2, AxResource res2, boolean insertBefore)
It moves the *res1* before/after *res2* whose AxmedisObject parent is specified by *parentOfRes2*. In a common list of resources the AxmedisObject parent is the root object and then the object itself. The *insertBefore* flags allows inserting before (*true*) or after

array getContent()
It returns an array of AxResource and AxmedisObject content or null

removeContent(AxResource res)
It deletes an existing AxResource content instance

removeContent(AxmedisObject res)
It deletes an existing AxmedisObject content instance

boolean setProtectionInfo(string toolID, number order, array parameters)
It allows setting the Protection Information Parameter according to the *toolID* (protection algorithm), the *order* of application of protection when multiple protection algorithms are applied and the *parameters* required by the algorithm. In the following table, details about available algorithm are shown:

toolID	***Description***	***parameters (Items of Array)***
0003	Performs encryption and decryption using Caesar cipher	int16 *offset*
0004	Performs encryption and decryption using Blowfish	0. string *key* (key of dec/enc) 1. string *iv* (initialization vector) 2. int16 *keyLength* 3. int16 *ivLength*
0005	Performs encryption and decryption using AES	
0006	Performs encryption and decryption using 3DES	
0007	Performs encryption and decryption using CAST-128	

Note: using the toolId it is necessary to apply the URN prefix "`urn:axmedis:ipmp:tool:id:`" as shown below:

```
var obj = new AxmedisObject();
...
//Applying Cipher algortihm to obj
var params = new Array();
params[0] = 20; //offset
var order = 1;
var toolId = "urn:axmedis:ipmp:tool:id:0003"; //prefix and tool code
obj.setProtectionInfo(toolId,order,params);
```

Please refer to the Cryptlib manual for further implementation details. (www.cryptlib.com).

boolean load (string AXOID, int version, boolean lock, string AXDBLoadEndPoint,string AXDBUser, string AXDBPasswd, string AXDBLockUnloadEndpoint)
Load method with AXMEDIS Database parameters. The Axmedis Object is loaded by using the *AXOID*, the *version*. The *lock* flag when *true* allows locking the object. The *version* set to -1 allows retrieving the last version of the object.

boolean load(string path)
It loads the axmedis object from filesystem

boolead save(string path)
It saves the axmedis object onto filesystem

boolean uploadToDB()
It uploads the axmedis object into the default database specifed into the configuration

boolean uploadToDB(string saverEndPoint, string user, string passwd,
boolean usingftp, string externalurl, string fileName, string internalurl,string lockEndPoint)
It uploads the axmedis object into a specified AXMEDIS database with a specified *fileName*

obtainDefinitiveAXOID(string AXCSObjeRegEndpoint, string AXCSObjeRegUsr,string AXCSObjeRegPsw)
The AXMEDIS Object is univocally defined by asking for a Definitive AXOID to the AXCS. AXCS to be used is defined by the location of server (*AXCSObjeRegEndpoint*), an authorised username (*AXCSObjeRegUsr*) and password (*AXCSObjeRegPsw*)

booelan registerToAXCS(string AXCSObjeRegEndpoint, string AXCSObjeRegUsr,string AXCSObjeRegPsw)
The AXMEDIS Object is registerd to AXCS by defining the location of server (*AXCSObjeRegEndpoint*), an authorised username (*AXCSObjeRegUsr*) and password (*AXCSObjeRegPsw*)

dispose()
The AXMEDIS Object is closed and destroyed to release memory. Only root objects are disposable.

JSAxInfo
Class name in the Javascript is *AxInfo*

It maps and allows managing the metadata of the AXINFO in JavaScript. This class manages the access to individual elements and fields in AXINFO metadata, this class maps all the functionalities provided by AxInfo class exposing setter and getter methods for accessing to data. It allows to manage:
• ObjectCreator information
• Owner information
• Distributor information
• Object Status information
• PromoOf information
• Workflow information
• Fingerprints information
• PAR information

Exposed properties

//Contibutor
number objectContributorCount
the number of ObjectContributor present

//Owner section
string ownerID
the code identifying the owner

string ownerIDCoding
coding used to identify the owner

string ownerName
the name of the owner

stringownerURL
URL of the owner

string ownerCompany
company of the owner

string ownerCompanyUrl
company URL of the owner

string ownerNationality
nationality of the owner

// Distributor section
boolean hasDistributor
True if a Distributor is present

string distributorAXDID
the AXIDID of the current distributor

string distributorName
the Name of the current distributor

string distributorURL
the URL of the current distributor

string distributorNationality
the nationality of the current distributor

// Object Status section
string objectAccessMode
the access mode: “READ_ONLY” or “READ_WRITE” string.

Date objectCreationDate
the local date and time of object creation

Date objectLastModificationDate
the local date and time of object modification

number objectVersion
version of the object

number objectRevision
revision of the object

string objectStatus
current status of the object

string objectType
object type (“BASIC” or “COMPOSITE”)

string objectProtectionStamp
the protection stamp

// PromoOf section
number promoOfAXOIDCount
count of AXOID in the PromoOf section

// Workflow section
string workflowItemID
WorkflowWorkItemID

string workflowWorkspaceInstanceID
WorkflowWorkspaceInstanceID

// Internal PAR
boolean hasInternalPAR
how many Internal PAR sections are present (0 or 1)

string internalPARStatus
the internal PAR status

// PAR section
boolean hasPAR
how many PAR sections are present (0 or 1)

string PARStatus
the PAR status

string PARLicensingURL
the licensing URL

// Status section
boolean isProtected
if the object is protected or not. True means protected.

boolean isGoverned
if the object contains a licence or not.

Exposed methods

AxInfo()
Empty contructor

AxInfo(AxInfo info)
Copy constructor. It create a copy of info object

// ObjectContributor Management
number addObjectContributor(number pos)
Adds a new ObjectContributor 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.

removeObjectContributor (number pos)
Removes an ObjectContributorfrom the position specified

number getObjectContributorCount()
Returns the number of ObjectContributor present

number findObjectContributorByAXCID(string axcid)
Returns the position of an ObjectContributor with a specific AXCID. It returns -1 if not found.

number findObjectContributorByName(string nam)
Returns the position of an ObjectContributor with a specific Name. It returns -1 if not found.

string getObjectContributorAXCID(number refNum)
allow getting the AXCID value for an ObjectContributor identified by position refNum

setObjectContributorAXCID(number refNum, string axcid)
allow setting the the AXCID value for an ObjectContributor identified by position refNum

string getObjectContributorName(number refNum)
allow getting the Name value for an ObjectContributor identified by position refNum

setObjectContributorName(number refNum, string name)
allow setting the Name value for an ObjectContributor identified by position refNum

string getObjectContributorURL(number refNum)
allow getting the URL value for an ObjectContributor identified by position refNum

setObjectContributorURL(number refNum, string URL)
allow setting the URL value for an ObjectContributor identified by position refNum

string getObjectContributorCompany(number refNum)
allow getting the Company value for an ObjectContributor identified by position refNum

setObjectContributorCompany(number refNum, string company)
allow setting the Company value for an ObjectContributor identified by position refNum

string getObjectContributorCompanyURL (number refNum)
allow getting the CompanyURL value for an ObjectContributor identified by position refNum

setObjectContributorCompanyURL(number refNum, string URL)
allow setting the CompanyURL value for an ObjectContributor identified by position refNum

string getObjectContributorNationality(number refNum)
allow to get and set the Nationality value for an ObjectContributor identified by position refNum

setObjectContributorNationality(number refNum, string nationality)
allow setting the Nationality value for an ObjectContributor identified by position refNum

// Owner Management>
boolean hasOwner()
True if a owner is defined

boolean removeOwner()
True if a owner is defined

setOwnerID(string value)
allow setting the code identifying the owner

number addOwnerDescription(number pos)
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(number pos)
removes the description specified

string getOwnerDescription(number pos)
allow getting the value of the description

setOwnerDescription(number pos, string description)
allow setting the value of the description

string getOwnerDescriptionLanguage(number pos)
allow getting the value of the description language

setOwnerDescriptionLanguage(number pos, string description)
allow seeting the value of the description language

// Distributor Management
removeDistributor()
removes the Distributor

// Object Status
string getObjectStatus()
allow getting current status of the object, the status values are factory dependent and set by the workflow therefore cannot be defined a priori.

setObjectStatus(string value)
allow setting the current status of the object, the status values are factory dependent and set by the workflow therefore cannot be defined a priori.

setObjectIsGoverned(boolean value)
allow setting if the object contains a licence or not. The license is not stored in the axinfo, the setter should be used to update the axinfo when the licence is added/removed from the object

// Object Creator Management
string getObjectCreatorAXCID()
allow getting the AXCID value for the ObjectCreator

setObjectCreatorAXCID(string val)
allow setting the AXCID value for the ObjectCreator

string getObjectCreatorName()
allow getting the Name value for the ObjectCreator

setObjectCreatorName(string name)
allow setting the Name value for the ObjectCreator

string getObjectCreatorURL()
allow getting the URL value for the ObjectCreator

setObjectCreatorURL(string url)
allow setting the URL value for the ObjectCreator

string getObjectCreatorCompany()
allow getting the Company value for the ObjectCreator

setObjectCreatorCompany(string company)
allow setting the Company value for the ObjectCreator

string getObjectCreatorCompanyURL()
allow getting the CompanyURL value for the ObjectCreator

setObjectCreatorCompanyURL(string URL)
allow setting the CompanyURL value for the ObjectCreator

string getObjectCreatorNationality()
allow getting the the Nationality value for the ObjectCreator

setObjectCreatorNationality(string nationality)
allow setting the Nationality value for the ObjectCreator

// PromoOf Management
addPromoOfAXOID(string axoid, number position)
adds a new AXOID in the PromoOf section, the position indicates where to put the AXOID, -1 means at the end

removePromoOfAXOID(number position)
removes the AXOID in the position specified

number getPromoOfAXOIDCount()
get the count of AXOID in the PromoOf section

string getPromoOfAXOID(number position)
allow getting the AXOID in a specified position

setPromoOfAXOID(number position, string value)
allow setting the AXOID in a specified position

// Internal Potential Available Rights Management
addInternalPotentialAvailableRights()
adds a new Internal PAR section

removeInternalPotentialAvailableRights()
removes the Internal PAR section

// Potential Available Rights Management
addPotentialAvailableRights()
adds a new PAR section if not present

boolean removePotentialAvailableRights()
removes the PAR section

setPotentialAvailableRights(string value)
allow setting the PAR string

JSAxResource
Class name in the Javascript is *AxResource*

The class models the AxResource type of the Axmedis Framework, it is used to store digital resource (audio, video, image, text, etc…) both raw data and Axmedis digital component. This class wraps the AxResource class and provides functionalities exposing setter and getter methods to:
• access to the mime type
• access to the byte stream of the resource
• create a new resource and to embed a file or a reference inside a resource object

Exposed attributes

string mimeType
The MimeType associated with the extension of digital resource

string ref
reference

string contentID
content identifier

string encoding
encoding type

string localPath
resource identifier

Exposed methods

AxResource ()
Default constructor

load (string path)
It loads a Raw Resource from filesystem

save (string path)
It saves a Raw Resource onto filesystem

JSAxDublinCore
Class name in the Javascript is *AxDublinCore*

Exposed properties

string metadataID

Exposed methods

addDCElement(string type, string value, string language)
Add a new element specifying type, value and language

addDCElement(string type, string value)
Add a new element specifying type, value. The language has an empty value

removeDCElement(string type, number refNum)
Delete an element specifying type and reference number.

string getDCElementValue(string type, number refNum)
Return an element specifying type and reference number.

string getDCElementValue(string type)
Return an element specifying type. The reference number is 0

setDCElementValue(string type, number refNum, string value)
Modify the element specifying type, reference number and value

string getDCElementLanguage(string type, number refNum)
Return the language of the element specifying type and reference number.

string getDCElementLanguage(string type)
Return the language of the element specifying type. The reference number is 0

setDCElementLanguage(string type, number refNum, string language)
Modify the language of element specifying type, reference number and language

number getDCElementCount(string type)
Return the number of elements specifying type.

JSAxMetadata
Class name in the Javascript is *AxMetadata*

Exposed properties

string namespace
The used namespace in the XML description. It is a readonly attribute.

Exposed methods

AxMetadata ()
Empty constructor

AxMetadata (string xml)

Constructor with parameter. It builds an object storing the XML passed as a string

addXML (string xml)

It stores an XML description of metadata passed as a string

string getXML ()

It returns the string with XML of the metadata description

11.3.3 Examples of usage

```
// Function for creating the Dublin Core information
function createDC(obj,title)
{
    dc = obj.getDublinCore();
    dc.addDCElement("creator","AXCP Rule Editor");
    dc.addDCElement("title",title);
    dc.addDCElement("type","conversion to "+format);
    dc.addDCElement("description","Testing JSScript with resized and converted images");
}

function adaptAxImages(axobj,height,width,mimeType)
{
  print("Creating adapted image");
  var h1 =  new Array(1);
  var w1 =  new Array(1);
  h1[0]=0;
  w1[0]=0;
  var resource = axobj.getContent();
  var i = 0;
  for (i in resource)
  {
    var res = resource[i];
    val = resource[i].mimeType.search("image");
       if(val==0)
    {

        ImageProcessing.GetInfo(resource[i],h1,w1);
        print("Original resource size is "+h1[0]+"x"+w1[0]);
        print("Resizing the resource at "+height+"x"+width);
        ImageProcessing.Resize(resource[i],height,width,false,resource[i]);

        ImageProcessing.GetInfo(resource[i],h1,w1);
        print("New resource size is "+h1[0]+"x"+w1[0]);
        if(resource[i].mimeType != format)
        {
            print("Converting the resource in "+format);
            ImageProcessing.Conversion(resource[i],mimeType,resource[i]);
        }
    }
  }
}

function cloneAxmedisObject(sourceObj,targetObj)
{
        var metadata = sourceObj.getGenericMetadata();
        if(metadata!=null)
        {
            for(j in metadata)
                targetObj.addMetadata(metadata[j]);
        }
        targetObj.addMetadata(sourceObj.getAxInfo());
        var resources = sourceObj.getContent();
        if(resources!=null)
        {
            for(k in resources)
               targetObj.addContent(resources[k]);
        }
}

// Function for creating the Axmedis Object by composing and converting resources
function test()
```

```
{
  var imgFormat = getMimeType(format);
  selection.resolveQuery("test","test",0);
  var documentList = selection.getAXDBResult();

  for (i in documentList)
  {

        uri ="axdb://"+documentList[i];
        var axmedisObject = new AxmedisObject(uri);
        var dublinCore = axmedisObject.getDublinCore();
        var title = dublinCore.getDCElementValue("title")+"_Resized";
        var newObj = new AxmedisObject();
        cloneAxmedisObject(axmedisObject, newObj);
        createDC(newObj,title);
        adaptAxImages(newObj,h,w,imgFormat);
        print("Storing Object on disk");
        newObj.save(resourcePath+title+".axm");

  }
  return true;

}

//Entry point of the script
test();
```

11.3.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section “AXMEDIS DATA Types and Functions for JavaScript”.

11.4 JSAXCPPlugin for AXMEDIS_CONTENT_PROCESSING Plugins (DSI)

Module Profile	
JSAXCPPlugin	
Responsible Name	Ivan Bruno
Responsible Partner	DSI
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Module
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/contentprocessing/ https://cvs.axmedis.org/repos/Framework/source/jsaxclasses/contentprocessing/
Reference to the AXFW location of the demonstrator executable tool for internal	

download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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,

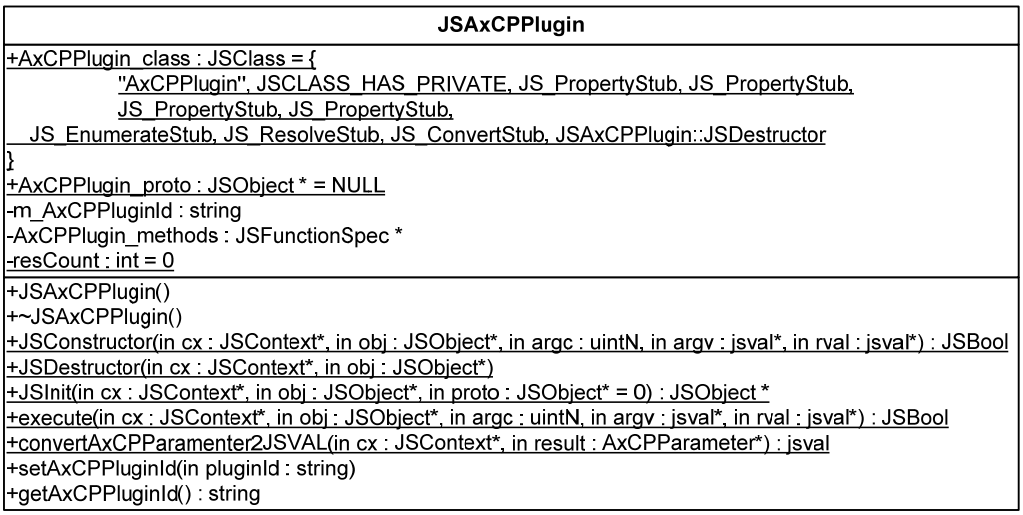
		proprietary, authorized or not
wxWidget	wxWidgets 2.4.2	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL Licence

JSAXCPPlugin is a metaclass that refers and wraps dynamically AXMEDIS Content Processing Plugins. It provides and manages the access to the set of functions exposed by DLLs discovered by the Plugin Manager and related to AXMEDIS content production, protection, adaptation algorithms and tools.
Dependencies of the rule define the number of and what plugins have to be loaded at the start of the execution, each plugin is a global instance of the AxCPPlugin class and the name of the instance is the same of the plugin.

Functions provided by the Axmedis Plugins for content processing

- *Fingerprint functions*
- *Digital Resource Adaptation functions*
- *Protection functions*
- *Metadata Adaptation functions*
- *Functions for using External tools*

11.4.1 Module Design in terms of Classes



11.4.2 Draft User Manual

JSAxCPPlugin
The Class name in Javascript is AxCPPlugin. This class is instatiated every time a dependency is defined in the rule. The allocation is perfomed automatically by the engine, the user can use such insance directly by means the name of the dependency (plugin) as an object. The class does not provide any attributes, but it is populated at runtime with methods that map the plugin functions. So that, dot notation is available for invoking such functions and each function can be call by passing the input/ouput parameters. The signature of the function is specified in the plugin profile XML description, see the example for more details.

11.4.3 Examples of usage

In the example below, the excerpt of JS script shows how to use a function of the ImageProcessing Plugin

[.....]

```
var imgResource = new AxResource()
```

```
imgResource.load(resourcePath+resName);
if(imgResource.mimeType.search("image")==0)
{
    ImageProcessing.Conversion(imgResource,format, imgResource);
    imgResource.save(outputPath+title);
}
```

[.....]

In the excerpt, the "resourcePath", the "outputPath", "res" and "format" are parameters of the AXCP Rule and they contain actual values respectively stating for:

- `resourcePath` is the path on the filesystem where find the resource
- `outputPath` is the path on the filesystem where store the resource
- `resName` is the filename of the resource
- `format` is the output format (i.e. expressed as mimetype)

Finally, `title` is a variable defined in the script at runtime used to set the output name of the resource
For more details see also the example in the JSAXOM section.

11.4.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section “AXMEDIS DATA Types and Functions for JavaScript”.

11.5 JSConnection classes (DSI)

Module Profile	
JSConnetction Classes	
Responsible Name	Ivan Bruno
Responsible Partner	DSI
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Module
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/.....
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.
Reference to the AXFW location of the demonstrator	

executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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
Wsdlpull	Wsdlpull 1.11	LGPL Licenze
LibCurl	Latest stable	LGPL license

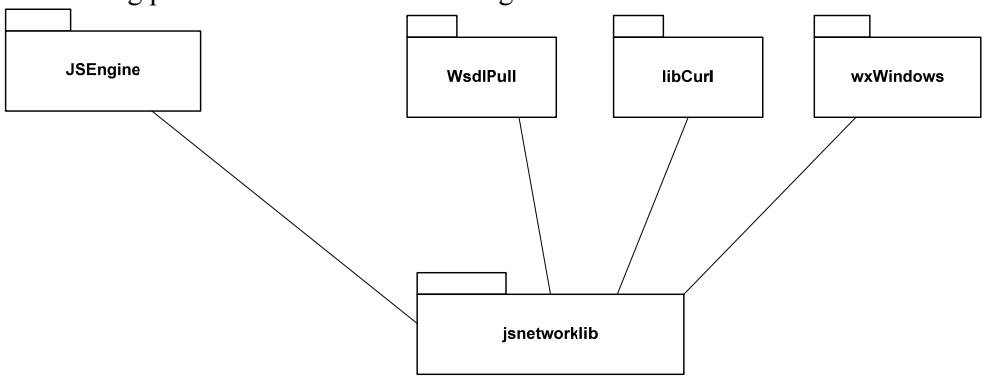
wxWidget	Latest stable	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL Licence

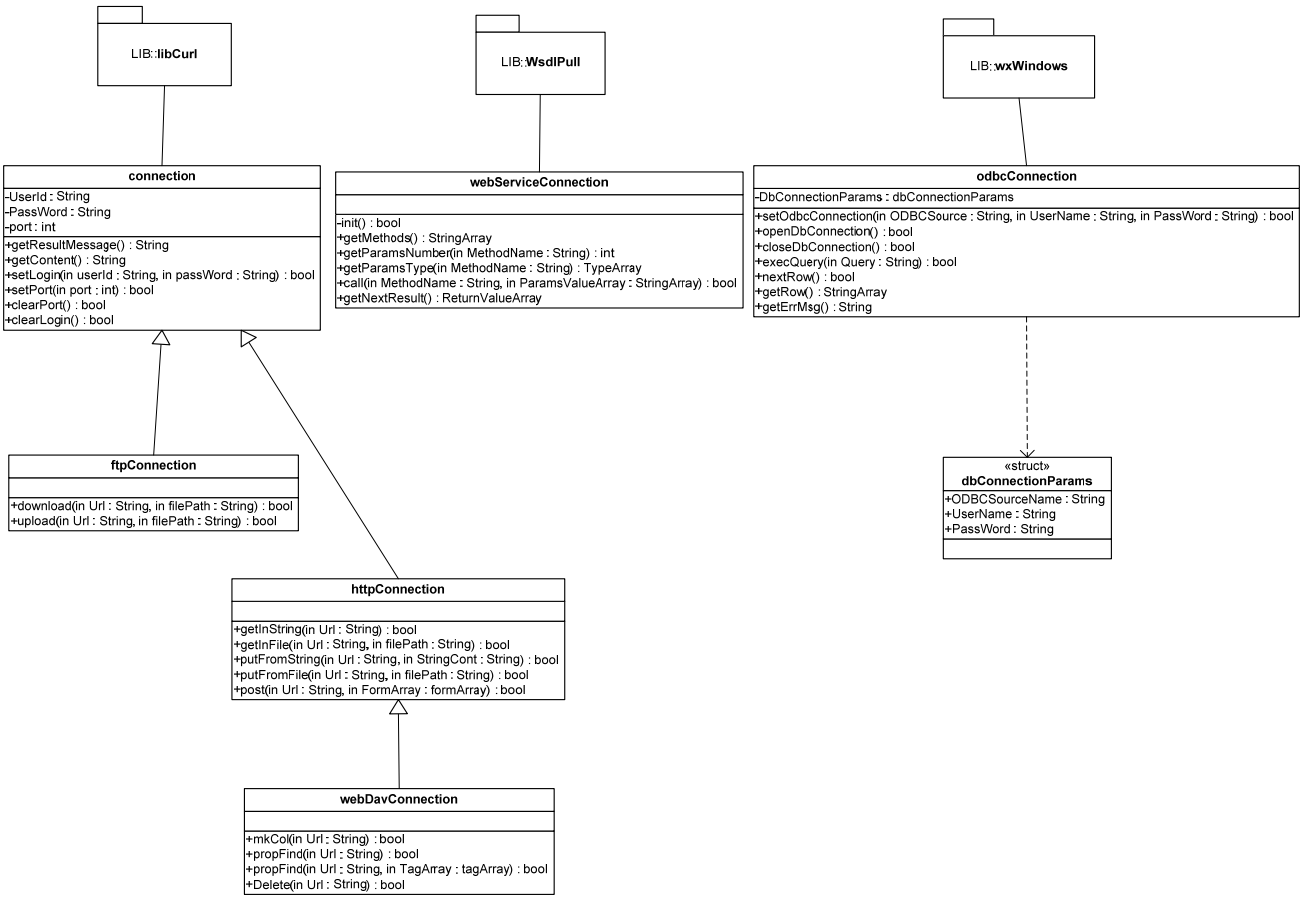
JS connection classes are a set of Javascript classes that provide the communiction support to the AXCP Engine by modeling in terms of primitive functionalities the network and database access. They consist in:

- **JShttpConnection** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSftpConnection** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSodbcConnection** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSwebServiceConnection** The class models a meta class for managing web services by loading WSDL description and dynamically creating services (methods).
- **JSForm** The class models a simple form of a http web page

11.5.1 Module Design in terms of Classes

In this section C++ connection classes are described. For each of them attributes and main methods are reported. Empty constructors and distructors are not listed, constructors that need input parameters are described. The following pictures describe the class diagrams.





Connection Class

This C++ class models a generalised network connection.

Attributes

- *UserId:string* – It is the username for login.
- *PassWord:string* – It is the password for login
- *Content:string* – It is the buffer where data is stored
- *Port:int* – It is the port number to use for the connection if it is different form the default port.

Methods

- *getResultMessage():String* – it provides the result of the last operation. In event of success an *Exit Succes* is returned otherwise the message string provides the type of fault.
- *setLogin(UserId:String,PassWord:String):bool* - It allows setting usernam and password for a login request.
- *getLogin():bool* – It tests the login parameters. If they were set the method returns TRUE, FALSE otherwise.
- *clearLogin():bool* – It clears the exisitng login parameters.
- *setPort(port:int):bool* – It allows settino a port different from the default.
- *clearPort():bool* metodo - It clears the port number resuming the default port.

ftpConnection class

The ftpConnection specialises the Connection class It allows performing download and upload file via ftp. It is necessary to provide the Url of the server where put/get the resource.

Attributes

No attributes

Methods

- *download(Url:String,FilePath:String):bool* – It performs the download of a file. The location of file is provided by the *Url* parameter, the file is stored on file system at the *FilePath* location.
- *upload (Url:String,FilePath:String):bool* – It allows uploading a file from the *FilePath* locationon own file system into a *Url* of the ftp server.

httpConnection Class

The httpConnection specialises the Connection class It allows using the access on network by using the http protocol.

Attributes

No attributes

Methods

- *getInString(Url:String):bool* – It allows performing the GET command from the specified *Url*. Received data can be retrieved by using the *getContent()* method.
- *getInFile(Url:String,FilePath:String):bool* – It allows performing the GET command from the specified *Url*. Received data are written directly on a file specified by *FilePath* parameter.
- *putFromString(Url:String,Buffer:String):bool* – It allows performing the PUT http command to the specified *Url.* The *buffer* parameter contains data to be sent to the server.
- *putFromFile(Url:String,FilePath:String):bool* – It allows performing the PUT http command to the specified *Url.* The *FilePath* parameter contains the location of file to be sent to the server.
- *post(Url:String,PostContent:FormArray):bool* – It allows performing the POST http command to the specified *Url*. We suppose to use this command to send to the server contents of some form-fields inside a html page. Before using this command it is necessary to create an array of Form object to use with the POST command.

Form Class

This class models a form in terms of name, type and value.

Attributes

- *formName:String* – It is the name of the form
- *formType:String* – It is the type of form
- *formValue:String* – It is the value of form to post

WebDavConnection class

The WebDavConnection specialises the httpConnection class It allows using the access on network by using the main methods of the WebDav protocol.

Attributes

No attributes

Methods

- *mkCol(Url:String):bool* – It allows invoking the MKCOL Web-Dav. It creates a collection of resources at the specified *Url*.
- *Delete(Url:String):bool* – It allows invoking the DELETE Web-Dav. It deletes the resource in the specified *Url*.
- *propFind(Url:String,TagArray:StringArray):bool* – It allows invoking the PROPFIND WebDav. It retrieves properties of the resource associated with the specified *Url*. It is possible to query specific properties by listing them in *TagArray.* Invoking the methods without an empty *TagArray* returns all properties.

OdbcConnection Class

The OdbcConnection class allows accessing to a database via Odbc.

Attributes
No attributes

Methods

- *setDbConnection(OdbcSource:String, UserName:String, PassWord:String):bool* – It allows setting the connection parameters. It has to be called before opening a Database connection.
- *openDbConnection():bool* – It opens a Database connection after the call of *setDbConnection*.
- *execQuery(query:String):bool* – It allows making a SQL query. Results are provided in a table and can be retrieved by calling the access methods: *nextRow* and *getRow*
- *nextRow():bool* – It allows browsing the table of results row by row. It set the internal pointer to the current row.
- *getRow():StringArray* – It allows retrieving content from the last pointed row. Values are returned in a array of string. The lenght of the array depends on the number of column of the table.
- *getErrorMessage():String* – It returns the last error message.
- *closeDbConnection():bool* – It closes the Database connection.

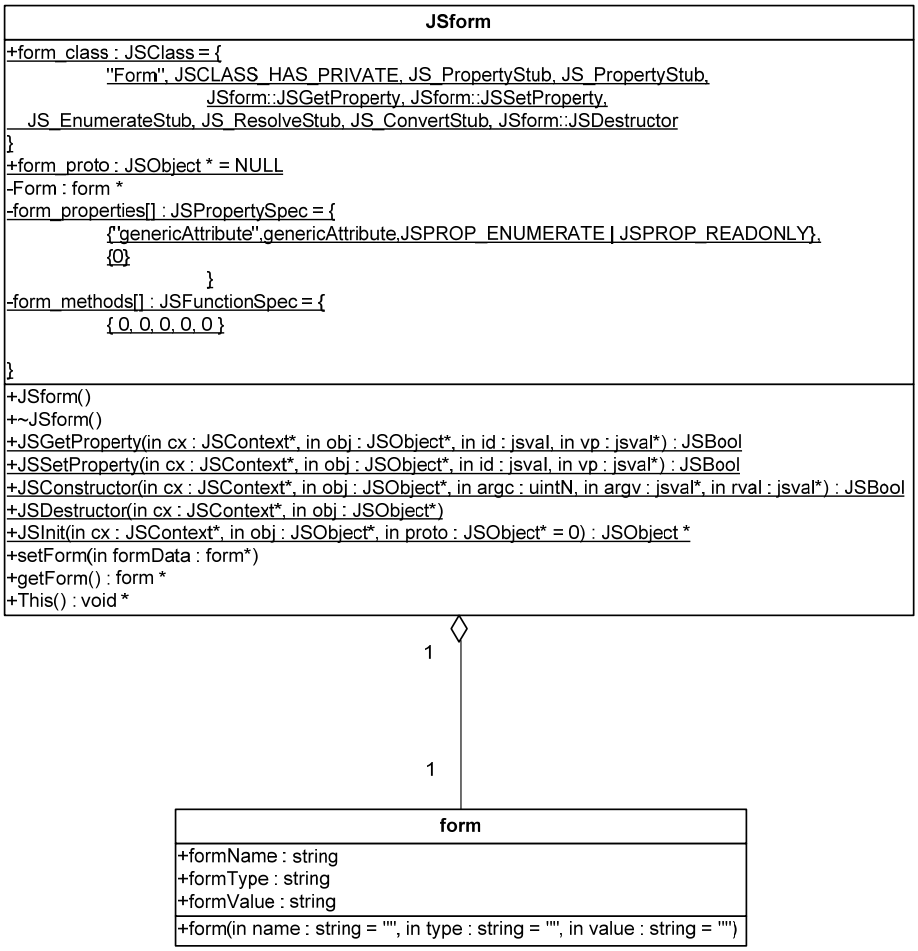
WebServiceConnection Class

The class allows opening a WebService connection based on a Wsdl. It creates at run-time a WebService client that allows invoking operations defined by the wsdl. Each operation is called by providing input paramters and return a result as operation return value.

Attributes
No attributes

Methods

- *WebServiceConnection(WsdlSource:string)* – Constructor that create a WebServiceConnection object based on the specified *WsdlSource*. The source can be a file or Url
- *getMethods():StringArray* – It returns an array where are stored the name of available operations.
- *getParamsNumber(MethodName:String):int* – It allows getting the number of reuired parameter for the operation specified in the *MethodName*.
- *getParamsType(MethodName:String):TypeArray* – It allows getting types of required parameters by the operation specified in the *MethodName*. Types are available in the *TypeArray* object
- *call(MethodName:String,Params:StringArray):bool* – The method allows invoking an operation on WebService. The *MethodName* specifies the operation to call with the list of *Params.*
- *getResult():ValuesArray* – The method gets the result returned by the call. The rusult is stored in an array of Value (*ValuesArray*).


JSform
+form_class : JSClass = {
"Form", JSCLASS_HAS_PRIVATE, JS_PropertyStub, JS_PropertyStub,
JSform::JSGetProperty, JSform::JSSetProperty,
JS_EnumerateStub, JS_ResolveStub, JS_ConvertStub, JSform::JSDestructor
}
+form_proto : JSObject * = NULL
-Form : form *
-form_properties[] : JSPropertySpec = {
{"genericAttribute",genericAttribute,JSPROP_ENUMERATE | JSPROP_READONLY},
{0}
}
-form_methods[] : JSFunctionSpec = {
{ 0, 0, 0, 0, 0 }
}
+JSform()
+~JSform()
+JSGetProperty(in cx : JSContext*, in obj : JSObject*, in id : jsval, in vp : jsval*) : JSBool
+JSSetProperty(in cx : JSContext*, in obj : JSObject*, in id : jsval, in vp : jsval*) : JSBool
+JSConstructor(in cx : JSContext*, in obj : JSObject*, in argc : uintN, in argv : jsval*, in rval : jsval*) : JSBool
+JSDestructor(in cx : JSContext*, in obj : JSObject*)
+JSInit(in cx : JSContext*, in obj : JSObject*, in proto : JSObject* = 0) : JSObject *
+setForm(in formData : form*)
+getForm() : form *
+This() : void *
1
1
form
+formName : string
+formType : string
+formValue : string
+form(in name : string = "", in type : string = "", in value : string = "")

11.5.2 Draft User Manual

This is a draft User Manual for the JS classes.

JSForm

This class wraps the *Form* class. Class name in Javascript is *Form*

Exposed Properties

string formName
: It is the name of the form

string formType
: It is the type of form

string formValue
: It is the value of form to post

Exposed methods

No methods

JShttpConnection

The class wraps the *httpConnection class.* Class name in Javascript is *HttpConnection*

Exposed methods

HttpConnection()
Create a new empty *HttpConnection* object

string getResultMsg()
Return the last Result Message: an *Exit Succes* is returned otherwise the message string provides the type of fault.

string getContent()
Return the html page as string

setLogin(string username, string password)
Set the account parameter for login

boolean getLogin()
Get the status of Login parameters, it returns *true* if they are set, *false* otherwise.

clearLogin()
Reset the account parameters for login

setPort(string port)
set a Port at the *port* number value

clearPort()
Clear the port number resuming the default port.

getToString(string url)
Get function, copy content at the specified *url* to a String. It is used in conjuction with *getContent()*.

getToFile(string url, string filePath)
Get function, copy content at the specified *url* to a File at *filePath*

putFromFile(string url, string filePath)
Put function, put a content From a File at *filePath* to a specified *url*

putFromString (string url, string buffer)
Put function, put content From a String *buffer* to a specified *url*

post(string url, array jsform)
Post at the specified *url* the array of *JSForm* data It allows performing the POST http command to the specified *Url*. We suppose to use this command to send to the server contents of some form-fields inside a html page. Before using this command it is necessary to create an array of JSForm object to use with the POST command.

JSftpConnection

Class name in Javascript is *FtpConnection*

Exposed methods

FtpConnection ()
Create a new empty *FtpConnection* object

string getResultMsg()
Return the last Result Message: an *Exit Succes* is returned otherwise the message string provides the type of fault.

string getContent()
Return the content as string

setLogin(string username, string password)
Set the account parameter for login

boolean getLogin()
Get the status of Login parameters, it returns *true* if they are set, *false* otherwise.

clearLogin()
Reset the account parameters for login

setPort(string port)
set a Port at the *port* number value

clearPort()
Clear the port number resuming the default port.

download(string Url,string FilePath)
It performs the download of a file. The location of file is provided by the *Url* parameter, the file is stored on file system at the *FilePath* location.

upload(string Url,string FilePath
It allows uploading a file from the *FilePath* locationon own file system into a *Url* of the ftp server.

JSodbcConnection

Class name in Javascript is *OdbcConnection*

Exposed methods

OdbcConnection()
Create a new empty *OdbcConnection* object

setDbConnection(string OdbcSource, string UserName, string PassWord)
It allows setting the connection parameters. It has to be called before opening a Database connection.

openDbConnection()
It opens a Database connection after the call of *setDbConnection*.

execQuery(string query)
It allows making a SQL query. Results are provided in a table and can be retrieved by calling the access methods: *nextRow* and *getRow*

nextRow()
It allows browsing the table of results row by row. It set the internal pointer to the current row.

Array getRow()
It allows retrieving content from the last pointed row. Values are returned in a array of string. The lenght of the array depends on the number of column of the table.

String getErrorMessage()
It returns the last error message.

closeDbConnection()
It closes the Database connection.

JSwebServiceConnection

Class name in Javascript is *WebServiceConnection*

Exposed Properties

wsdlURI
It is the URI used to load the WSDL

Exposed methods

WebServiceConnection(string URI)
Costructor method loads the WSDL specified by the URI string input parameter.

All other methods are defined dynamically according to the loaded WSDL,
All input parameters are string.

11.5.3 Examples of usage

```
var ftp = new FtpConnection();
ftp.setLogin (UserName, PassWord);
ftp.download (URL, DestinationPath);
print(ftp.getResultMsg());

var http = new HttpConnection();
var cont ent = new string();
http.setPort(80);
http.getToString(URL);
content = http.getContent();
print(content);

var http = new HttpConnection();
var formArray = new Array(2);
var content = new String();
formArray[0] = new Form(”FormName”, ”FormType”, ”FormValue”);
formArray[1] = new Form(”FormName”,”FormType” ,”FormValue”);
http.post(URL, formArray);
content = http.getContent();
print(http.getContent());

var webDav = new WebDavConnection();
webDav.mkCol(URL);
content = webDav.getContent();
print(content);

var odbc = new OdbcConnection();
var RowArray = new Array(3) ;
odbc.setOdbcConnection(DbName, UserName, PassWord);
```

```
odbc.openDbConnection();
odbc.execQuery(QUERY) ;
RowArray[0] = odbc.getRow();
odbc.nextRow();
RowArray[1] = odbc.getRow();
print(odbc.getErrorMsg());
odbc.closeDbConnection();

var Array = new Array();
var WebService = new WebServiceConnection(URL);
Array = WebService.generateAxoid(”mario ”, ”xxxxx”, ”Pluto”);
```

11.5.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section “AXMEDIS DATA Types and Functions for JavaScript”.

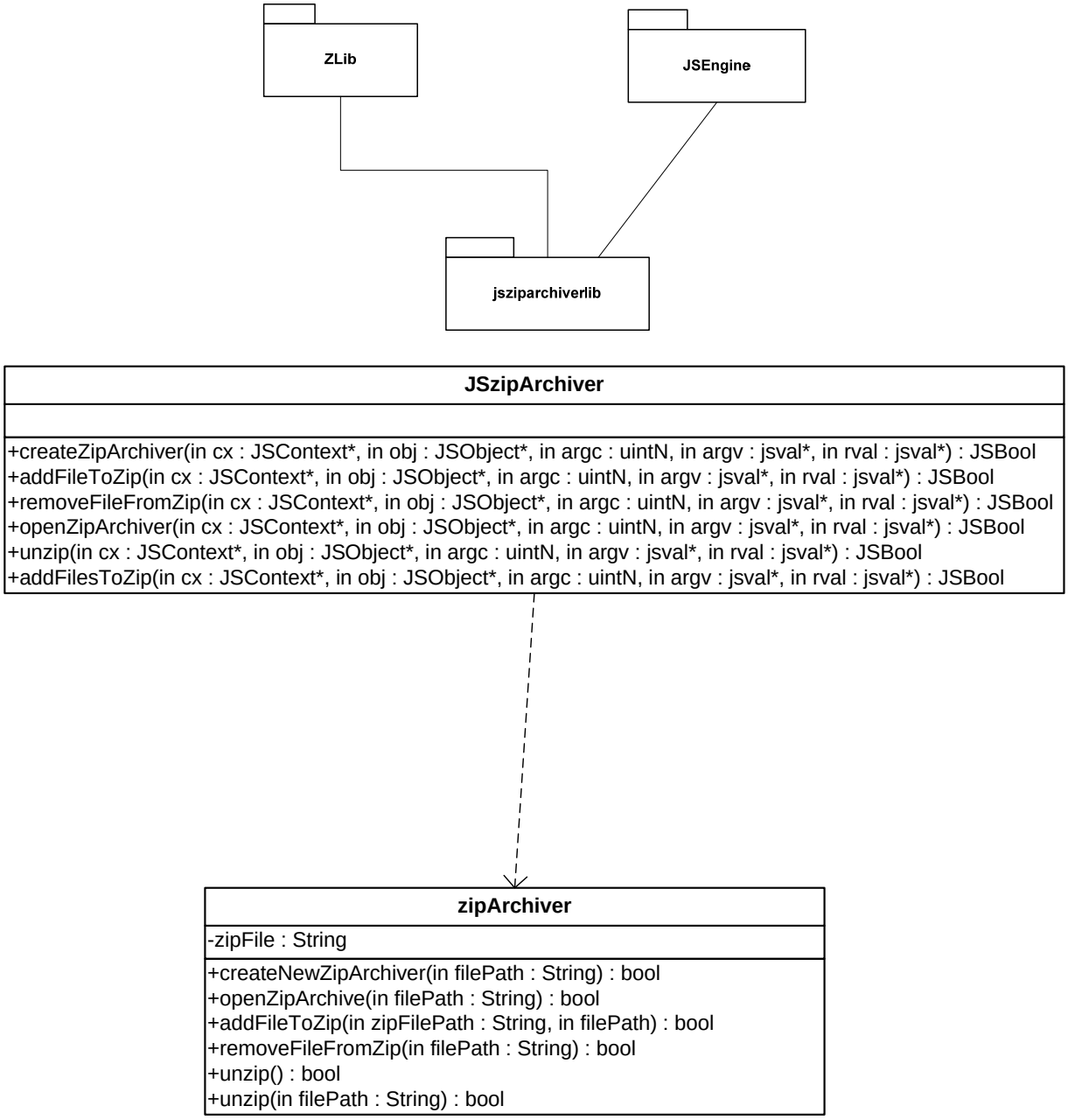
11.6 JSZipArchiver class (DSI)

Module Profile	
JSZipArchiver Classes	
Responsible Name	Ivan Bruno
Responsible Partner	DSI
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Module
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/.....
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.
Reference to the AXFW location of the demonstrator executable tool for public download	
Address for accessing to WebServices if any, add accession information (user and Passwd) if any	
Test cases (present/absent)	

Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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	Latest stable	ZLIB license
JS API Spidermonkey	JS API v.1.5	LGPL Licence

JS_ZipArchiver class models Javascript class for creating archives of files based on the ZIP algorithm.

11.6.1 Module Design in terms of Classes



ZipArchiver Class

Attributes

zipFile: string

The string contains the file as array of chars

Methods

- *createNewZipArchive(filePath:String):bool* – It creates a new zip file at the filePath location. filePath has to contain the name of file.
- *openZipArchive(filePath:String):bool* – It opens a existing zip archive at the filePath location.
- *addFileToZip(zipFilePath:String, filePath:String):bool* - It adds a file at the *filePath* location to the *zipFilePath* archive.
- *removeFileFromZip(fileName:String):bool* – It deletes a *fileName* file from the archive.

- *unzip():bool* – It allows to unzip the archive in the same folder of the archive.
- *unzip(dirPath:String):bool* – It allows to unzip the archive in the *dirPath* folder.

11.6.2 Draft User Manual

This is a draft User Manual for the JS classes.

JSzipArchiver
Class name in Javascript = *ZipArchiver*

Exposed methods

ZipArchiver()
It creates *a ZipArchiver* object

createNewZipArchive(string filePath)
It creates a new zip file at the filePath location. filePath has to contain the name of file.

openZipArchive(string filePath:String)
It opens a existing zip archive at the filePath location.

addFileToZip(string zipFilePath, string filePath)
It adds a file at the *filePath* location to the *zipFilePath* archive.

removeFileFromZip(string fileName)
It deletes a *fileName* file from the archive.

unzip()
It allows to unzip the archive in the same folder of the archive.

unzip(string dirPath)
It allows to unzip the archive in the *dirPath* folder.

11.6.3 Examples of usage

```
var Zipper = new ZipArchiver();
Zipper.createZipArchive (”C:/pippo.zip”);
Zipper.addFileToZip (”C:/pluto.txt”);

var Zipper = new ZipArchiver();
var FilesList = new Array();
FilesList[0]=”C:/pippo.txt”;
FilesList[1]=”C:/minnie.txt”;
Zipper.addFilesToZip (”pippo zip”, FilesList);

var Zipper = new ZipArchiver();
Zipper.openZipArchive(”C:/pippo.zip”);
Zipper.unzip();
```

11.6.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	

Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section "AXMEDIS DATA Types and Functions for JavaScript".

11.7 JSAxSelection (DSI)

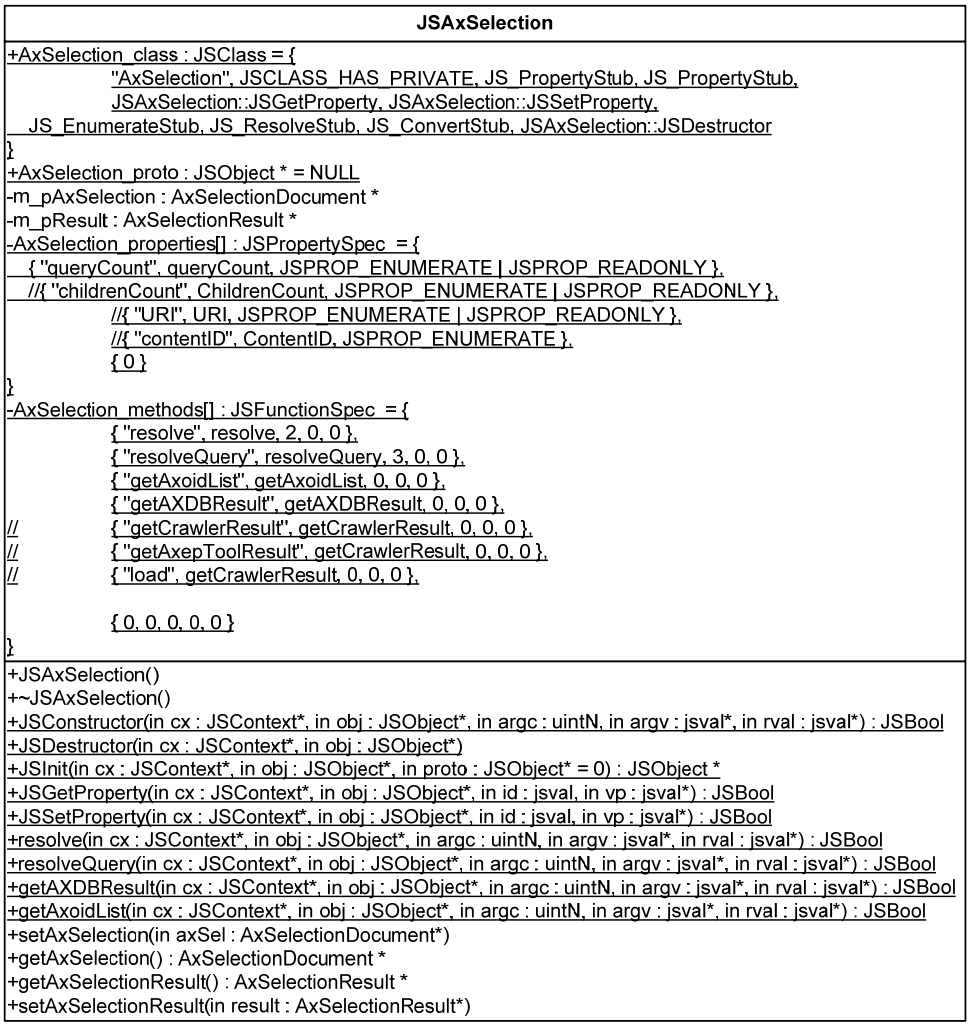
Module/Tool Profile		
JSAxSelection		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)	Approved	
Implemented/not implemented	Implemented	
Status of the implementation	70%	
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows, linux and Mac OSx	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/selection/ https://cvs.axmedis.org/repos/Framework/source/jsaxclasses/selection/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.axmedis.org/repos/Applications/ruleeditor/bin/win32/ https://cvs.axmedis.org/repos/Applications/ruleeexecutor/bin/win32/	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user aNd Passwd) if any	https://cvs.axmedis.org/repos/WebServices/QuerySupportWS/ https://cvs.axmedis.org/repos/WebServices/UserSelectionArchive/	
Test cases (present/absent)	Absent	
Test cases location		
Usage of the AXMEDIS configuration manager (yes/no)	Yes	
Usage of the AXMEDIS Error Manager (yes/no)	No	
Major Problems not solved	Wsdlpull library is not in final release, so changes in it impacts on selection document classes	
Major pending requirements		
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)

Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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	Xerces 2.6.0	LGPL
WSDLPULL	Wsdlpull 1.11	LGPL
WxWidgtes	wxWidgets 2.4.2	LGPL
JS API Spidermonkey	JS API v.1.5	LGPL Licence

JSAxSelection class maps the Selection document in the JavaScript. This class allows using Selection objects to manage the access and to make queries to the AXMEDIS database, and to retrieve AXMEDIS objects ID (AXOID). It manages the array of AXOID provided by the MainQuerySupport when the selection is resolved to be actualized.

11.7.1 Module Design in terms of Classes

The JSAxSelection wraps the *AxSelectionDocument* class referring to it by means of the *m_pAxSelection* pointer.



11.7.2 Draft User Manual

This is a draft User Manual for the JSSelection class.

JSAxSelection
Class name in Javascript = *AxSelection*

The AxSelection class allows managing selection XML data structure. It provides access to the Main Query Support Web Service in order to query for Axoids by means complex set of queries.

Exposed properties:

number queryCount
It provides the number of queries in the selection *(read only)*

string querySupportURI
It provides the URI of the Main Query Support Web Service

string archiveURI
It provides the URI of the Selection Archive Web Service

Exposed methods:

AxSelection(...)
Default constructor.

AxSelection(string xmlString)
Constructor with parameter. It builds the selection instance by using the xml description stored in *xmlString*

resolve(string user, string pwd)
The method resolve the whole selection. The *user* and *pwd* are needed to access to Main Query Support service

resolveQuery(string user, string pwd, int queryIndex)
The method resolve the query specified by the *queryIndex*. The *user* and *pwd* are needed to access to Main Query Support service

array getAXDBResult()
It returns an array of axoids related to Axmedis Objects stored into the Axmedis DB and retieved by the last query or selection.

array getP2PResult()
It returns an array of axoids related to Axmedis Objects available into AXEPTOOLS (P2P) and retieved by the last query or selection.

array getCMSResult()
It returns an array of axoids related to media objects available into CMS and retieved by the last query or selection.

boolean load(string filepath)
The method load the selection from the File System specified by the *filepath*.

boolean loadFromDB(string user, string pwd, int selId)
The method load the selection from the Selection Archive specified by the *selId*. The *user* and *pwd* are needed to access to Main Query Support service

string getXMLQuery(number queryIndex)
It returns the xml of a query by means the *queryIndex*

boolean addXMLQuery(string xmlQuery)
It adds the xml of a query to selection.

boolean removeQuery(number queryIndex)
It deletes the query specified by the *queryIndex*

11.7.3 Examples of usage

The following example shows the usage of a *selection* object. The *selection* instance is a global object that manages a selection parameter provided by the AXCP Rule. In the example, the selection is used to

actualised the first query (‘0’). The `resolveQuery` method allows to call the MainQuerySupport and `getAXDBResult` method provides the list of AXOIDS returned as result (Actualisation).

```
function test()
{
  var imgFormat = getMimeType(format);

  //actualisation of selection
  selection.resolveQuery("test","test",0);

  //getting the AXOIDs
  var documentList = selection.getAXDBResult();

  for (i in documentList)
  {
     uri ="axdb://"+documentList[i];
     var axmedisObject = new AxmedisObject(uri);
     var dublinCore = axmedisObject.getDublinCore();
     var title = dublinCore.getDCElementValue("title")+"_Resized";
     var newObj = new AxmedisObject();
     cloneAxmedisObject(axmedisObject, newObj);
     createDC(newObj,title);
     adaptAxImages(newObj,h,w,imgFormat);
     print("Storing Object on disk");
     newObj.save(resourcePath+title+".axm");
  }
  return true;
}

//Entry point of the script
test();
```

11.7.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section “AXMEDIS DATA Types and Functions for JavaScript”.

11.8 AXSBJS - AXMEDIS searchbox javascript bridge (DSI with focuseek)

Module Profile	
AXSBJS - AXMEDIS searchbox	
Responsible Name	Baldini
Responsible Partner	Focuseek
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Module

Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/crawler/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		

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
expat	Latest stable	Expat license
EasySOAP++	Latest stable	LGPL license
wxWidget	Latest stable	LGPL license
zlib *JS API Spidermonkey*	Latest stable JS API v.1.5	ZLIB license LGPL Licence

11.8.1 General Description of the Module

It is the AXMEDIS searchbox javascript bridge, developed by focuseek for use with the SpiderMonkey JavaScript Engine.

11.8.2 Module Design in terms of Classes

The searchbox bridge provides the following set of JS classes:

- **AXSearchbox:** The main object for accessing searchbox.
- **Document**: Used to handle documents as opaque objects.
- **MetadataValue**: Used to hold metadata.
- **QueryParser**: Used to specify the query string parser to use.
- **QueryInfo**: Used to specify the information returned as result of a query.
- **QueryView**: Used to restrict the set of documents returned as result of a query.
- **QueryAtomType:** Used to specify the type of a QueryAtom.
- **QueryAtom:** Used to build a query in RPN notation.
- **QuerySliceWeight:** Used to specify slice weights.
- **QuerySpec:** Used to submit a query.
- **QueryResult:** Used to return information on a query result

These classes are available in the AXSBJS DLL that is registered via the AXSBJSInit() function.

11.8.3 Draft User Manual

The AXSBJS DLL provides a single entry point, called AXSBJSInit(), it must be used during the SpiderMonkey initialization phase. This function registers the following classes that are required to communicate with the searchbox engine.

AXSearchbox
The main object for accessing searchbox.
Class name in the Javascript is *AXSearchbox*

Exposed Properties:

- host [string] - Host where searchbox server is running.
- port [string] - Port where searchbox server is listening.
- username [string] - Username for authentication.
- password [string] - Password for authentication.

Exposed Methods:

- integer query(in QuerySpec query, out QueryResults[] results) - Performs the query specified by query and returns matching documents into results. The maximum number of results is returned.
- string addDocument(in integer arcid, in string url, in Document doc) - Adds a document to the specified arcid. The URL of the document is specified in url and the document in doc. The ID of the document is returned. The caller must have write access rights for the archive where the document is stored.
- void removeDocument(in string docid) - Removes a document from the archive it belongs. The ID of the document is specified in docid. The caller must have write access rights for the archive where the document is stored.
- Document getDocument(in string docid) - Returns the cached copy of the page with specified docid. The document is returned. The caller must have read access rights for the archive where the document is stored.
- string getDocumentFFF(in string docid) - Returns the FFF representation of the page with specified docid. The FFF XML is returned. The caller must have read access rights for the archive where the document is stored.
- DocumentMetadata getDocumentMetadata(in string docid) - Returns an object that describes the metadata associated to the document (either applied via MetadataTemplates, or stored in the FFF). The caller must have read access rights for the archive where the document is stored.
- string getDocumentURL(in string docid) - Returns the URL of the document docid. The caller must have read access rights for the archive where the document is stored.
- string normalizeURL(in string url) - Returns the normalized URL.
- void applyMetadataTemplate(in string docid, in integer templateid, in MetadataValue[] metadata) - Applies a MetadataTemplate to a document, with the specified variable metadata. The caller must have write access rights for the archive where the document is stored.
- void deapplyMetadataTemplate(in string docid, in integer templateid) - Deapplies a metadata template from a document. The caller must have write access rights for the archive where the document is stored.
- integer[] enumAppliedMetadataTemplates(in string docid) - Returns the IDs of the applied MetadataTemplates for the specified document. The caller must have read access rights for the archive where the document is stored.

Document
Used to handle documents as opaque objects.
Class name in the Javascript is *Document*

Exposed Properties:
- mimeType [string] - MIME type of the document.
- size [integer] - size (in bytes) of the document.
- Methods:
- void read(in string filename) - Reads document from file.
- void write(in string filename) - Writes document to file.
- void readFromBuffer(in integer address, in integer size) - Reads a document from a memory buffer at the specified address with the specified size.
- void writeToBuffer(in integer address, in integer size) - Writes a document in a memory buffer at the specified address with the specified maximum size.

MetadataValue
Used to hold metadata.
Class name in the Javascript is *MetadataValue*

Exposed Properties:
- key [string] - Metadata key.

- slice [integer] - Slice where metadata is stored.
- value [string] - Metadata value.

DocumentMetadata
Used to describe the metadata associated with a document.
Class name in the Javascript is *DocumentMetadata*

Methods:
- string getValue(in string key) - Gets the first value of the metadata with the specified key. Returns and empty string if the metadata was not found.
- string[] getValues(in string key) - Gets all the values of the metadata with the specified key.
- MetadataValue getMetadataValue(in string key) - Gets the first MetadataValue object for the metadata with the specified key. Returns null if the metadata was not found.
- MetadataValue[] getMetadataValues(in string key) - Gets all the MetadataValue objects for the metadata with the specified key.
- MetadataValue[] getAllMetadataValues(in string key) - Gets all the MetadataValue objects for every metadata of the document.

QueryParser
Used to specify the query string parser to use.
Class name in the Javascript is *QueryParser*

Static Exposed Properties:
- NOPARSER - Don't parse, the query is submitted using QueryAtoms.
- RPNPARSER - Use the RPN parser.
- ALGPARSER - Use the ALG parser.
- NETPARSER - Use the NET parser.

QueryInfo
Used to specify the information returned as result of a query.
Class name in the Javascript is *QueryInfo*

Static Exposed Properties:
- INFO_NONE - For each result no additional info is returned.
- INFO_URL - For each result the URL is returned.
- INFO_TITLE - For each result the URL and the title is returned.
- INFO_CONTEXT - For each result the URL, the title, the mime type and the contexts where the keywords specified into the query have been found are returned.
- INFO_TEMPLATE_METADATA - For each result the URL, the title, the mime type, the contexts where the keywords specified into the query have been found and the metadata added by templates to the document are returned.
- INFO_ALL_METADATA - For each result the URL, the title, the mime type, the contexts where the keywords specified into the query have been found and all the metadata of the document are returned.

QueryView
Used to restrict the set of documents returned as result of a query.
Class name in the Javascript is *QueryView*

Static Exposed Properties:
- VIEW_PUBLISHED - The query is applied to all the documents currently in the archive.

- VIEW_CORECHANGED - The query is applied only to the documents currently in the archive that have changed in the core of the text. Only applicable to an historicizing archive.

QuerySort
Used to specify the sorting of documents returned as result of a query.
Class name in the Javascript is *QuerySort*

Static Exposed Properties:
- SORT_STANDARD - The standard sorting is used.
- SORT_RELEVANCE - The documents are ordered by relevance score.
- SORT_SCORE - The documents are ordered by their intrinsic score.
- SORT_TIME_NEWER - The documents are ordered by change timestamp, more recently changed documents first.
- SORT_TIME_OLDER - The documents are reverse-ordered by change timestamp, least recently changed documents first.

QueryAtomType
Used to specify the type of a QueryAtom.
Class name in the Javascript is *QueryAtomType*

Static Exposed Properties:
- ATOM_WORD - QueryAtom is a keyword to find.
- ATOM_WILDCARD_WORD - QueryAtom is a keyword with wildcards to find.
- ATOM_NOT - QueryAtom is a logic NOT.
- ATOM_AND - QueryAtom is a logic AND between other QueryAtoms.
- ATOM_OR - QueryAtom is a logic OR between other QueryAtoms.
- ATOM_NEAR - QueryAtom is logic NEAR between words.
- ATOM_META - QueryAtom is a meta-keyword to find.
- ATOM_META_RANGE - QueryAtom is a meta-keyword range to find.
- ATOM_WILDCARD_META - QueryAtom is a meta-keyword with wildcards to find.

QueryAtom
Used to build a query in RPN notation.
Class name in the Javascript is *QueryAtom*

Exposed Properties:
- type [integer] - Current QueryAtom type (from QueryAtomType).
- meta [string] - Contains the meta-keyword type. Only for META QueryAtoms.
- param [string] - If the current type is WORD or META it contains the keyword (or meta-keyword) to find. Otherwise, if the current type is AND, OR or NEAR it contains the decimal representation of the number of QueryAtom involved in the expression, and if the current type is META_RANGE it contains the lower boundary of the range. For the NOT type, only the value "1" is allowed in this field.
- param1 [string] - If the current type is META_RANGE it contains the upper boundary of the range. Otherwise, if the current type is NEAR it contains the decimal representation of the sloppyness allowed. An empty or 0 sloppyness makes the NEAR match an exact phrase, a sloppyness greater than 0 sets the maximum number of keyword swaps that can be made to match the query.

QuerySliceWeight
Used to specify slice weights.
Class name in the Javascript is *QuerySliceWeight*

Exposed Properties:

- slice [integer] - Dict ID. The following dict IDs can be used:
    0. Author
    1. Keyword
    2. Abstract
    3. Invisible
    4. Marginal normal text
    5. Marginal emphasized text
    6. Marginal link text
    7. Marginal remote link text
    8. Marginal header text
    9. Central normal text
    10. Central emphasized text
    11. Central link text
    12. Central remote link text
    13. Central header text
    14. Title
- weight [integer] - Slice weight. Must be greater or equal to 0.

QuerySpec
Used to submit a query.
Class name in the Javascript is *QuerySpec*

Exposed Properties:

- archives [array of integers] - IDs of the archives you want to query. Leave empty if you want to query a collection or a watch.
- collection [integer] - ID of the collection you want to query. If you want to query archives or a watch, use 0.
- watch [integer] - ID of the watch you want to query. If you want to query archives or a collection, use 0.
- firstDoc [integer] - Index of the first document (starting from 0) returned. It must be less than lastDoc.
- lastDoc [integer] - Index of the last document (starting from 0). It must be greater than firstDoc.
- minTime [integer] - Oldest Timestamp (expressed in number of seconds since January 1st 1970 GMT) of query results. All older documents will be rejected.
- maxTime [integer] - Newest Timestamp (expressed in number of seconds since January 1st 1970 GMT) of query results. All newer documents will be rejected.
- minScore [integer] - Minimum score of query results. All documents with lower score will be rejected.
- info [integer from QueryInfo] - Detail level of query results.
- view [integer from QueryView] - Document set restrictions of query.
- sort [integer from QuerySort] - Result document set sorting type.
- parser [integer from QueryParser] - Parser to use to parse the query string.
- query [array of QueryAtoms] - Query in RPN notation all list of QueryAtoms. The QueryAtom sequence must produce a stack with only one element. Only used if NOPARSER is specified as QueryParser.
- queryString [string] - Query string to be parsed. Only used if RPNPARSER, ALGPARSER or NETPARSER is specified as QueryParser.
- weights [array of QuerySliceWeights] - Slice weights. You can pass an empty vector to use the default slice weights. To disable a slice in the current query, you must pass an entry for the slice with a weight of 0. If you don't pass an entry for a certain slice, that slice will have its default weight.

QueryResult

Used to return information on a query result.
Class name in the Javascript is *QueryResult*

Exposed Properties:

- id [string] - ID of the document, guaranteed to be unique across archives.
- url [string] - URL of the document.
- title [string] - Title of the document.
- mimeType [string] - Mime type of the document.
- contexts [array of strings] - Contexts of the document where the keywords have been found.
- timestamp [integer] - Document timestamp (expressed in number of seconds since January 1st 1970 GMT).
- serverTime [integer] - Document timestamp (expressed in number of seconds since January 1st 1970 GMT), as reported by the server.
- score [integer] - Document score (expressed as percentage * 10000).
- archives [array of integers] - The archives the document belongs to.
- categories [string] - Private use.
- metadata [array of MetadataValues] - Metadata associated with the document.
- templates [array of integers] - IDs of the MetadataTemplates applied to the document.

11.8.4 User interface description

No specific user interface.

11.8.5 Technical and Installation information

The bridge comes in the form of a redistributable DLL, a stub library (MSVC71 format) for linking to the DLL, include files and dependancy DLL libraries. To install the bridge, copy the dlls somewhere in your path (preferably on the same directory where the main program that links to the library resides), and place stub library and include files in the correct place for your development system.

References to other major components needed	Collector Indexer
Problems not solved	none
Configuration and execution context	

11.9 JS_Protection (FHGIGD)

JS_Protection is covered by the protection related functions of JS_AXOM and JS_ProtectionViewer. They provide the protection methods, which are needed by the rule editor for the JavaScript.

The following functionalities are provided as a JavaScript stub by JS_AXOM and JS_ProtectionViewer using AXOM, ProtectionProcessor and PMS:

JS_AXOM:

- Adding the protection information to the Object properties.
- Protecting the Object based on Protection information added to the Object: encryption, scrambling, compression.

JS_ProtectionViewer

- Viewing available Protection Tool IDs and Names

The required keys are requested from PMS (encryption keys) and the protection information (keys and parameter, see IPMP standard of MPEG21) is sent to the database of the AXCS via the PMS.

Protection Information
The protection information holds the IPMP information as stated in MPEG-21 Part 4 IPMP standard and more. The functinality to modify the protection information of an AXMEDIS object is provided by JS_AXOM.

The protection information may include:

- How each element of an AXMEDIS object has been protected, i.e. encrypted, encoded, compressed and scrambled.
- The IDs of the Protecton Tool used for Protection
- The key use for the protection.
- The Order of the different protection steps if the object was protected by more than once by protection algorithms.
- It is based on an XML schema which allows to describe sort of protection procedures as en-/decryption, (de-)compression, (de-)scrambling

The syntax and semantics is are 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.

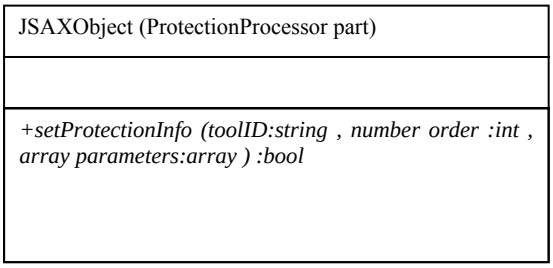
JS_ProtectionViewer
The following functionality is provided by JS_ProtectionViewero retrieve information about the available protection tools:

- Viewing the names of the available protection tools.
- Viewing the IDs of the available protection tools..

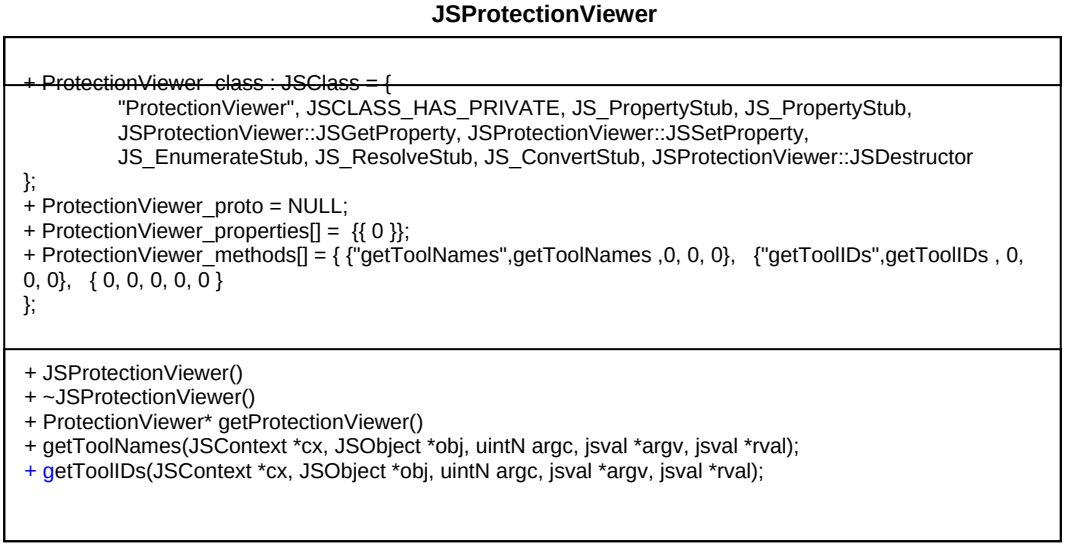
11.9.1 Module Design in terms of Classes

The following diagrams describe the design of the protection related modules. For more detailed information about the different protection algorithms, please take a look at the specification of the Protection Processor.

JS_AXOM
(simplified view showing the protection related functions)



JS_ProtectionViewer



11.9.2 Technical and Installation information

References to other major components needed	Protection Processor
Problems not solved	On-going work.
Configuration and execution context	JS_Protection is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section "AXMEDIS DATA Types and Functions for JavaScript".

11.9.3 Draft User Manual

This module is on-going work. A detailed user manual will be available when the work is completed.

JSAXObject

Exposed methods (related to protection)

boolean setProtectionInfo (string toolID, number order, array parameters)

It allows setting the Protection Information Parameter according to the *toolID* (protection algorithm), the *order* of application of protection when multiple protection algorithms are applied and the *parameters* required by the algorithm. In the following table, details about available algorithm are shown:

toolID	***Description***	***parameters (Items of Array)***
0003	Performs encryption and decryption using Caesar cipher	int16 *offset*
0004	Performs encryption and decryption using Blowfish	0. string *key* (key of dec/enc)
0005	Performs encryption and decryption using AES	1. string *iv* (initialization vector)
0006	Performs encryption and decryption using 3DES	2. int16 *keyLength*

JSProtectionViewer

Exposed methods

string getToolNames (void)
This method returns names of available protection tools.

string getToolIDs(void)
This method returns the IDs of available tools.

11.9.4 Examples of usage

```
// Adding the Protection information and Applying Protection

var obj = new AxmedisObject();
//Applying Cipher algortihm to obj
var params = new Array();
params[0] = 20; //offset
var order = 1;
var toolId = "urn:axmedis:ipmp:tool:id:0003"; //prefix and tool code
obj.setProtectionInfo(toolId,order,params);
obj.save();

 // Viewing the information about available protection tools.

var protectionviewer_1 = new ProtectionViewer();
rval= "\n protectionviewer_1 \t ToolNames:"+ protectionviewer_1.getToolNames() +
"\n protectionviewer_1 \t ToolIDs:"+ protectionviewer_1.getToolIDs();
return true;
```

11.9.5 Integration and compilation issues

Please refer to section "AXMEDIS DATA Types and Functions for JavaScript".for information about integration an compilation issues of JS modules within the AXCP Rule Editor and Engine.

11.10 JS_DRM (FHGIGD)

JS_DRM defines the data types and methods related to related to license and PAR for the AXCP Engine.

The following functionalities are provided within JS_DRM as a JavaScript stub using AXOM and PMS:

- Creating a new governed object (with license)
- Sending a License to the PMS and saving the license into the database
- Loading a License from the database via the License ID
- Loading a License Model from the database via the License Model ID
- Generating a license from license model and additional information (principal, AXOID)
- Check/Verification of an issued License against some RIGHTS written in clear such as: "the play on the AXOID 34 in July 2005 for 5 times, the print of AXOID 56 in Spain in May 2006 at least one, etc."
- Check/Verification if it is possible to issue/generate a License with some RIGHTS written in clear such as: "the play on the AXOID 34 in July 2005 for 5 times, the print of AXOID 56 in Spain in May 2006 at least one, etc."
- Check/Verification of existing PAR against some RIGHTS written in clear such as: "the play on the AXOID 34 in July 2005 for 5 times, the print of AXOID 56 in Spain in May 2006 at least one, etc."
- Addition of rights or removal from a license (license adaptation): Generation of a new license (with new or less rights) AND Revocation of the old licenses in ONE TRANSACION
- Addition of rights or removal from a PAR (PAR adaptation): Generation of a new PAR (with new or less rights) AND Revocation of the old licenses in ONE TRANSACION
- Check/Verification of license against PAR
- Rights expressrion translation between OMA and MPEG21
- Generation of new license from available license patterns
- Verification of the license according to the PAR and parent licenses

- Verification that the license generated by the user fulfills the initial requirements of the user. For example, the user can verify that with this license he could exercise the desired action over the AXObject.

In order to express DRM rules associated to AXMEDIS objects it has been decided to use MPEG-21 REL as primary rights expression language. A common structure is imposed for licenses and PAR.

JS_License – class that models a license

The following objects are needed to fully represent a *license* and all of its components in JavaScirpt:

- JS_License
- JS_Issuer
- JS_GrantGroup
- JS_Grant
- JS_Right
- JS_Principal
- JS_Resource
- JS_Condition
    - Fee
    - Territory
    - Number
    - Interval

In addition to the above classes following objects are used to exercise license creation and management within the AXCP engine:

- JS_LicensePattern
- JS_Verification
- JS_Certification
- JS_ActionLog
- JS_CondiotionsandMore
- JS_RightsExpressionTranslator
- JS_OMALicense

Each *license* has an *issuer* and a *GrantGroup*. Each *GrantGroup* contains a set of *Grants*. Each *Grant* contains the information of the *right* granted, the *resource*, the *principal* and an optional set of *conditions* related to that right. In addition, we have to realise that a *resource* can be a *GrantGroup* (in case of Distributor Licenses).

For expressing the different types of conditions JS_CONDITION, the following information was defined:

- ConditionType: It indicates which kind of condition we are expressing.
- Five Tvalue fields and two NValue fields (more can be added if desired), whose values depend on the conditionType.

JS_LicenseManager – class that models a license manager

This class provides the following functionality:

- Storing a License to the file
- Storing a License to Securecache
- Reteriving License from the file
- Removing the License
- Changing License Format
- Reteriving the PAR from the file
- Storing PAR.
- Removing PAR

JS_PMSClient – class that models a PMSClient
This class wraps the PMSClient class which provides the interface to the PMS. This class is responsible for the handling of license management related jobs amongst the different modules and the PMSServer.

In order to avoid two many parameters from the scripts the following JS clases are provided which represents set of parameters used in the webservice method call:

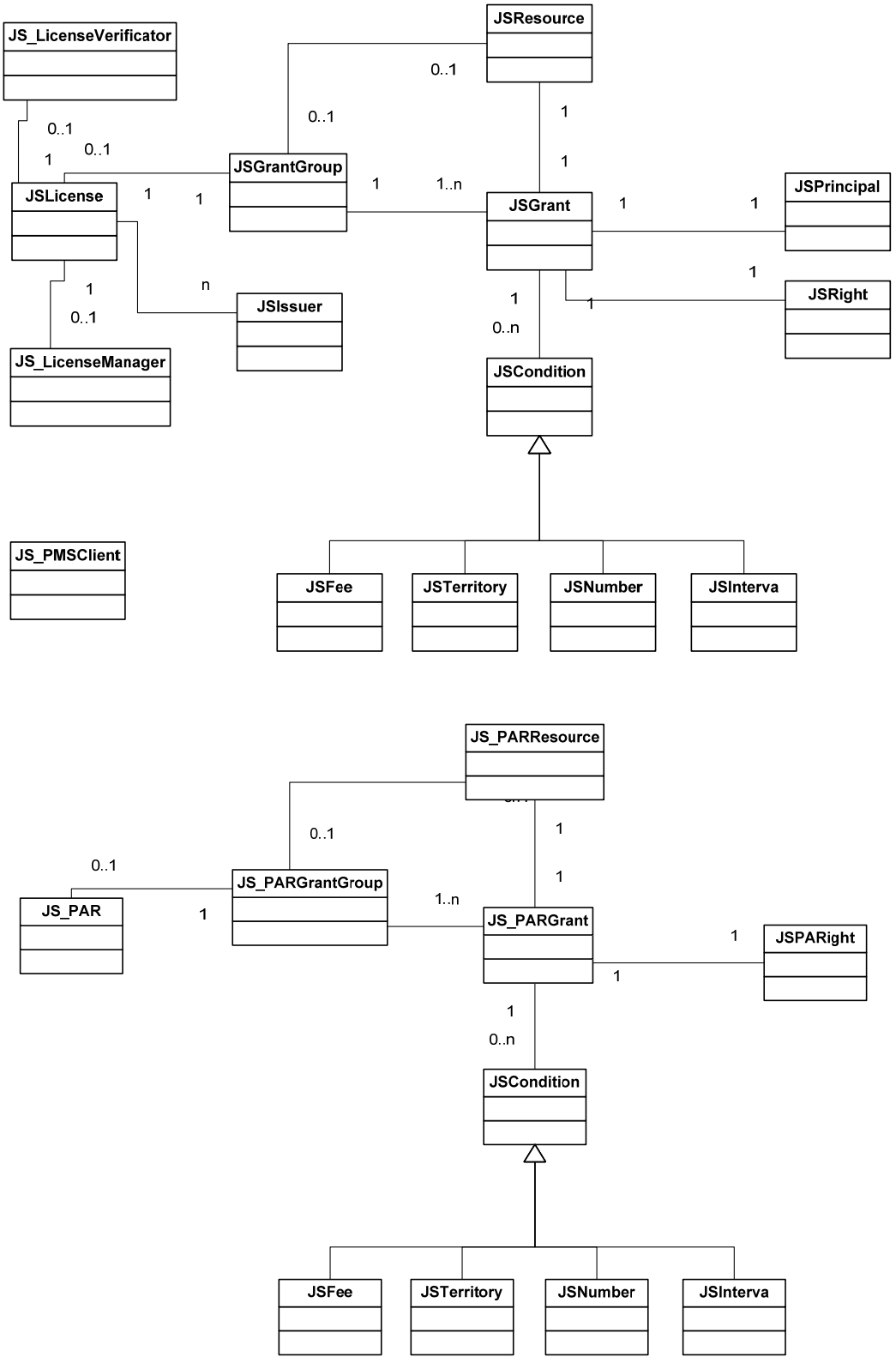
- JS_RightsParameters
- JS_RightsFeeParameters
- JS_RightsLimitParameters
- JS_RightsIntervalParameters
- JS_RightsRegionParameters
- JS_AXToolInfoParameters
- JSProtInfoParameters
- JS_OpInfoParameters
- JS_AXIDParameters
- JS_PARGrantParameters

JS_PAR – class that models a PAR
The following objects are needed to fully represent a *PAR* and all of its components in JavaScript:

- JS_PAR
- JS_ParGrant
- JS_ParRight
- JS_ParResource
- JS_ParGrantGroup
- JS_Condition
    - Fee
    - Territory
    - Number
    - Interval

The relationship between the different objects and the meaning is the same as for JS_LICENSE.

11.10.1 Module Design in terms of Classes



11.10.2 Technical and Installation information

References to other major components needed	PMS Server

Problems not solved	
Configuration and execution context	JS_Protection is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section "AXMEDIS DATA Types and Functions for JavaScript".

11.10.3 Draft User Manual

This module is on-going work. A detailed user manual will be available when the work is completed.

JS_License
License is the certificate provided to the user for particular content.

Exposed methods

addGrant(Grant g)
This method adds a grant to a grant group.

getIssuer(void)
This method returns the issuer of the license.

setIssuer(Issuer i)
This method establishes the issuer of the license.

getGrantgroup()
This method returns a copy of the grant group of the license.

setGrantGroup(GrantGroup gg)
This method establishes the grant group of the issueance.

getXMLLicense()
This method returns the license in XML string format.

sendLicenseToPMS(string endP,string clientcert,string passClientcert,string caCert)
This method sends the license to the PMS server.

RetrieveLicenseFromFile(string filepath)
This method retrieves the license from a file.

StoreLicenseToFile(string filepath)
This method stores the license in the file.

LicenseManager
LicenseManager is the class that manages The license database.

Exposed methods

StoreLicense(License lic)
This method stores the license.

StoreLicense(string lic)
This method stores the license. It is the overloaded method.
StoreLicense(License lic, string licenseDBid)

This method stores the license. it is the overloaded method which takes the two arguments mentioned above.

StoreLicenseInSecureCache(string lic, string licID)
This method stores the license in the secure cache.

StoreLicenseToFile(License &lic,string filepath)
This method stores the license to a file.

RetrieveLicenseFromFile(string filepath,int &errorcode)
This method retrieves the license from a file.

StringXMLLicense2LicenseModel(string XMLLicense,int &errorcode)
This method converts the license from a string to a license model.

RetrievePARFromFile(string filepath,int &errorcode)
This method retrieves the PAR from a file.

StringXMLPAR2PARModel(string XMLLicense,int &errorcode)
This method converts the license from a string to a licensemodel.

StorePAR(PAR parp)
This method stores the PAR.

StorePAR(string parp)
This method stores the PAR. It is an overloaded method.

RetrieveLicense(string licenseId)
This method retrieves the license.

RetrievePAR(string PARId)
This method retrieves the PAR.

*DataLicense(string licenseId,char **p,int *s);*
This method gets data from license.

LicenseManager StorePAR(string AXOID,string par)
This method stores the PAR.

LicenseManager StorePAR(string AXOID,PAR parp)
This method stores the PAR.

LicenseManager::RemovePAR(string AXOID)
This method removes the PAR.

RightseExpressionTranslator
This class provides access method for converting license formats.

Exposed methods

generateTranslation(string _license, string _originalRel, string _destinationRel)
This method converts teh "original license" from its REL to the destination one.

*setRight(char *n)*

This method establishes the right.

JS_PMSClient
A PMSClient is the client interface for the PMS. It is responsible for distributing all protection related jobs amongst the different modules and the PMSServer.

Exposed methods

string initLicenseEndUser(string IssuerAXUID)
This method initialises the creation of a license. This is the first Web service to be called in the process of an end user license creation. The service initLicenseEndUser returns the temporary identifier of the license. This identifier is usable while the license is being created. When the license is finished and stored this identifier is not used any more and it is deleted from the database.

addGrantEndUser(RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)
This is the Web service that adds (one each time) the rights granted in a license. This service has to be called as many times as rights granted by the license. The different parameters allow introducing: the right, the resource over which the right will be exercised, the user who will obtain the right, and finally, the different conditions to be accomplished.

finaliseLicenseEndUser(string licenseTmpId)
This method finalises the license. This is the last service to be invoked in a license creation process. The service builds the license and, if it is correct, then stores it in the database.

initLicenseDistributor(string IssuerAXUID)
This method initialises the creation of a license. This is the first Web service to be called in the process of a distributor license creation. This service receives information about the creator of the license. The service initLicenseEndUser returns the temporary identifier of the license. This identifier is usable while the license is being created. When the license is finished and stored this identifier is not used any more and it is deleted from the database.

addGrantforDistributor(RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)
This method is the Web service that adds (one each time) the different rights for distributors and the distribution conditions for each one. The parameters established in this service affect only to the issuer right (the one defining distribution). This service has to be called as many times as distributors the license has. The different parameters allow introducing: the distribution conditions, the content that will be distributed and the identification of the distributor.

addGrantforEndUser(RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)
This method is the service that adds (one each time) the rights that a distributor can distribute. In other words, this function adds the rights that can be included in an EndUser license created by a specific distributor. This service has to be called as many times as different rights will be available in the future EndUser licenses. The different parameters allow introducing: right and the different conditions to be accomplished. The resource is established before in the addGrantforDistributor service.

finaliseLicenseDistributor(licenseTmpId)
This method is a Web service which finalises the license. This is the last service to be invoked in a license creation process. The service builds the licenses and, if it is correct, then stores it in the database.

initPAREndUser()

This method is a Web service which initialises the creation of a PAR. This is the first Web service to be called in the process of an end user PAR creation.

addPARGrantEndUser(PARGrantParameters pgrantpar,RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)

This method is the Web service that adds (one each time) the different rights for distributors and the distribution conditions for each one. The parameters established in this service affect only to the issue right (the one defining distribution). This service has to be called as many times as distributors the license has. The different parameters allow introducing: the distribution conditions, the content that will be distributed and the identification of the distributor.

finalisePAREndUser(string PARTmpId)

This method is a Web service which builds the PAR and, if it is correct, then stores it in the database. This is the last service to be invoked in a PAR creation process.

initPARDistributor()

This method is a Web service which initialises the creation of a PAR. This is the first Web service to be called in the process of a distributor PAR creation. This service receives information about the creator of the license. The service returns the temporary identifier of the PAR. This identifier is usable while the PAR is being created. When the PAR is finished and stored this identifier is not used any more and it is deleted from the database.

addPARGrantforDistributor(PARGrantParameters pgrantpar,RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)

This method is the service that adds (one each time) the different rights for distributors and the distribution conditions for each one. The parameters established in this service affect only to the issue right (the one defining distribution). This service has to be called as many times as distributors the PAR has. The different parameters allow introducing: the distribution conditions, the content that will be distributed and the identification of the distributor.

addPARGrantforEndUser(PARGrantParameters pgrantpar,RightsParameters rpar,RightsIntervalParameters rintervalpar,RightsLimitParameters rlimitpar,RightsRegionParameters rregionpar,RightsFeeParameters rfeepar)

This method is the service that adds (one each time) the rights that a distributor can distribute. In other words, this function adds the rights that can be included in an end user PAR created by a specific distributor. This service has to be called as many times as different rights will be available in the future end user licenses. The different parameters allow introducing: right and the different conditions to be accomplished. The resource is established before in the addGrantforDistributor service.

finalisePARDistributor(string licenseTmpId)

This method is the service which finalises the license. This is the last service to be invoked in a PAR creation process. The service builds the PAR and, if it is correct, then stores it in the database.

getLicense(string licenseId)

The method getLicense retrieve a license from the database (locally or remotely).

sendLicense(string XMLLicense)

This method sends an XML license created with the AX Editor to the PMS Server. The server will verify that license and will store it.

getPAR(string PARId)

This method retrieve a PAR from the database (locally or remotely).

sendPAR(string XMLPAR)

The method sendPAR sends an XML PAR created with the AX Editor to the PMS Server. The server will verify that PAR and will store it.

authorise(AXIDParameters axidpar,OpInfoParameters opinfopar,ProtInfoParameters protinfopar) (

This method verifies and authorise that an action can be exercised.

certify(string axid, AXTOOLInfoParameters axpar, string axdom)

This method is used to certify a tool, user and device in the AXCS via the PMS Server. It first checks the user information and then, it performs some checks in the following order: whether tool is registered, tool status, tool registration deadline and tool fingerprint.

If all the above information is right, it creates a new entry in the CerTools table of the AXCS database and returns a {@link CertificationResult} struct with all its fields filled. Otherwise, it returns a {@link CertificationResult} struct with an error code that identifies what happened.

verify(string axid,AXTOOLInfoParameters axtoolinfopar)

This method is used to verify a tool, user and device against the AXCS via the PMS Server. It first checks the user information and then, it performs some checks in the following order: whether the tool is present in the CerTools table, tool status, tool user, tool domain, tool registration deadline, the hash of the tool fingerprint and tool operation history (LastFPPA) consistency. The latter check is performed through the AXMEDIS Supervisor.

reverify(string axid,AXTOOLInfoParameters axtoolinfopar)

This method is used to reverify a tool, user and device against the AXCS via the PMS Server. This method is intended to be used by the Protection Processor when the result of the verify method determines that the toolFingerprintDigest parameter did not match the expected one. In this case, the full tool Fingerprint must be sent instead of the Hash.

verifyUser(string axid, string axdom)

This method is used to verify a user against the AXCS via the PMS Server. Verifies if the user is registered is the specified domain (if present) and checks that the user status and registration deadline are valid, so that the user can still use the AXMEDIS tools and the AXMEDIS framework.

getProtectionInfoLOCAL(string object, string version, string protstamp)

This method is used to get the AXMEDIS protection information of a specific object from the AXCS Objects database in Local Secure Cache.

deleteProtectionInfo(string object, string version, string protstamp)

This method is used to get the AXMEDIS protection information of a specific object from the AXCS Objects database in Local Secure Cache.

updateProtectionInfoREMOTE(string id, ProtInfoParameters protinfopar)

This method is used to update the AXMEDIS protection information of a specific object in the Objects database in the AXCS (via the PMS Server). When someone calls this method passing the identifiers of one object and the new AXMEDIS protection information, the Supervisor accesses the AXCS database and updates AXMEDIS protection information of the corresponding object.

verifyPAR (string par)

This method is used to verify a PAR syntactically against the schemes defined within the PAR.

verifyTempDistLicenseAgainstPAR (string distrLicense, string par)

This method verifies if the license can be generated according to the PARs stored in a database and the parent license(e.g Distributor or Creator).

verifyTempDisLicenseAgainstPARDataBase (string distrLicense, string pardatabase, string pardatabasehost, string pardatabaseuser, string pardatabasepass)

This method verifies if the license can be generated according to the PARs stored in a Database and the parent licenses (e.g Distributor or creator licenses).

verifyLicense (string license)

The methods checks if a License XML string is well formed against schemas.

generateTranslation (string _license, string _originalRel, string _destinationRel)

This method will convert the original license from its REL to the destination one. The Input parameter "_originalRel" can only be either "MPEG_21" or "OMA".

11.10.4 Examples of usage

In the following a Javascript source code is reported as example of usage:

```
// License handling

// creating Issuer object
var myIssuer_1 = new Issuer();
myIssuer_1.setIssueTime("2006-01-01T20:15:00");
myIssuer_1.setDetails("Axmedis License");
myIssuer_1.setIssuer("Axmedis");
myIssuer_1.setId("4711");

// creating Grant Object
var grant1 = new Grant();
grant1.setId("Alice")
var c1 = new Fee();
var c2 = new LNumber();
var c3 = new Interval();
var c4 = new Territory();
c1.setAmount(12.32);
c2.setCount(3);
c3.setNotBefore("2005:12:31");
c4.setDomain("USA");
grant1.addCondition(c1);
grant1.addCondition(c2);
grant1.addCondition(c3);
grant1.addCondition(c4);

// creating Grant Object
var grant2 = new Grant();
grant2.setId("Bob");
var c5 = new Fee();
var c6 = new Interval();
var c7 = new LNumber();
var c8 = new Territory();
c5.setAmount(4.11);
c7.setCount(5);
c6.setNotBefore("2007:12:31");
c8.setDomain("GB");
grant2.addCondition(c5);
grant2.addCondition(c6);
grant2.addCondition(c7);
grant2.addCondition(c8);

// creating GrantGroup object, and adding created Grants
var myGrantGroup_1 = new GrantGroup();
myGrantGroup_1.addGrant(grant1);
myGrantGroup_1.addGrant(grant2);


/**
*
*   License
```

```
*
*/

// License constructor
var myLicense = new License();
myLicense.setIssuer(myIssuer_1);
myLicense.setGrantGroup(myGrantGroup_1);



// LicenseManager constuctor
var mylicensemanager_1 = new LicenseManager();
 var mypar = new PAR();
  var myLicense = new License ();
// setServiceReference(string servicereference)

// setCurrency(string currency)

// stringXMLLicense2LicenseModel(string xmllicense,int&errorcode)
// setAmount(float amount)
mylicensemanager_1.accessCache();
//
mylicensemanager_1.accessDB();
// DataLicense (string licenseid ,string par,int no)
 // mylicensemanager_1.DataLicense("license_id","p",0);
 mylicensemanager_1.SetConnectionsParams("DBhost","DBlicdatabase","DBuser","DBpassword");


// LicenseManager copy-constructor
 var mylicensemanger_2 = new LicenseManager(mylicensemanager_1);

rval = "\n licensemanger_1 \n\t license:
"+mylicensemanager_1.RetrieveLicenseFromFile("fielpath",1)+
" \n\t licensemangerinstance : " +mylicensemanager_1.getInstance()+
" \n\t license : " +mylicensemanager_1.RetrieveLicense ("license_id")+
" \n\t PAR : " + mylicensemanager_1.RetrievePAR("PAR_id");
 " \n\t storelicense return code : " + mylicensemanager_1.StoreLicense("license")+
 " \n\t storelicense return code : " + mylicensemanager_1.StoreLicense(myLicense_1
,"licenseDBId")+
 " \n\t storelicense return code : " + mylicensemanager_1.StoreLicense(myLicense_1 )+
 " \n\t storelicense return code : " + mylicensemanager_1.StoreLicenseToFile( myLicense
,"filepath");
 " \n\t storePAR return code : " + mylicensemanager_1.StorePAR("AXOID","PAR") +
 " \n\t storePAR return code : " +mylicensemanager_1.StorePAR(mypar,"PAR") +

"\n\t conditionsandmore: "+ mylicensemanager_1.getPosibleConditions("AXUID","Right","AXOID")+
" \n\t conditionsandmore: "+
mylicensemanager_1.getPosiblePARConditions("right","AXOID","PARdb","PARdbhost","PARdbuser","PARdb
pass")+
" \n\t conditionsandmore: "+
mylicensemanager_1.getPosibleConditionsOfIssue("AXLID","GGId_EndUserGrant","GId_EndUserGrant")+
" \n\t license: "+mylicensemanager_1.StringXMLLicense2LicenseModel("xmlLicense",0)+
" \n\t PAR: "+mylicensemanager_1.RetrievePARFromFile ("filepath",0)+
 " \n\t PAR: "+mylicensemanager_1.StringXMLPAR2PARModel("XMLPAR",1)+
  " \n\t removeparerrorcode : "+ mylicensemanager_1.RemovePAR("AXOID")+
  " \n\t storeLicenseInSecureCache return value : "+
mylicensemanager_1.StoreLicenseInSecureCache("License","licID");



// RightsExpressionTranslator  constuctor
var myrightsexptranslator_1 = new RightsExpressionTranslator ();

// generateTranslation

rval = "\nmyrightsexptranslator_1   \n\tTypeCondition:
"myrightsexptranslator_1.generateTranslation("license_1","license_2") ;



// PMSCLient constuctor
var mypmsclient_1 = new PMSClient();

// parameters for method addGrantEndUser
// rightsparameters class
var myrightsparameters_1 = new RightsParameters();
myrightsparameters_1.setLicenseTmpId("Temp license id");
```

```
myrightsparameters_1.setAXUIDPrincipal("Axuid principal");
myrightsparameters_1.setAXOID("AXOID");
myrightsparameters_1.setDiType(1);
myrightsparameters_1.setDiSubType(1);
myrightsparameters_1.setRight("right");
myrightsparameters_1.setAdaptationRules ("Adaptation rules");



// RightsIntervalParameters
var myrightsintervalParameters_1 = new RightsIntervalParameters();
myrightsintervalParameters_1.setValidityInterval (true);
myrightsintervalParameters_1.setNotBefore("nbefore");
myrightsintervalParameters_1.setNotAfter ("nafter");


//RightsLimitParameters
var myrightslimitparameters_1  = new RightsLimitParameters();
myrightslimitparameters_1.setCountLimit (true);
myrightslimitparameters_1.setLimit (1);

// RightsRegionParameters
var myrightsregionparameters_1 = new RightsRegionParameters();
myrightsregionparameters_1.setValidityRegion (true);
myrightsregionparameters_1.setCountry("Germany");
myrightsregionparameters_1.setRegion("EU");

// RightFee parameter class
var myrightsfeeparameters_1 = new RightsFeeParameters();
myrightsfeeparameters_1.setFee(1.23);
myrightsfeeparameters_1.setFeeType(1);
myrightsfeeparameters_1.setCurrency("EURO");
myrightsfeeparameters_1.setBankAccount("Deusche Bank");




// parameters for method addGrantDistributor

    // rightsparameters class
var myrightsparameters_2 = new RightsParameters();
myrightsparameters_2.setLicenseTmpId("Temp license id");
myrightsparameters_2.setAXUIDPrincipal("Axuid principal");
myrightsparameters_2.setAXOID("AXOID");
myrightsparameters_2.setDiType(1);
myrightsparameters_2.setDiSubType(1);



    // RightsIntervalParameters
var myrightsintervalParameters_2= new RightsIntervalParameters();
myrightsintervalParameters_2.setValidityInterval (true);
myrightsintervalParameters_2.setNotBefore("nbefore");
myrightsintervalParameters_2.setNotAfter ("nafter");


    //RightsLimitParameters
var myrightslimitparameters_2  = new RightsLimitParameters();
myrightslimitparameters_2.setCountLimit (true);
myrightslimitparameters_2.setLimit (1);

    // RightsRegionParameters
var myrightsregionparameters_2 = new RightsRegionParameters();
myrightsregionparameters_2.setValidityRegion (true);
myrightsregionparameters_2.setCountry("Germany");
myrightsregionparameters_2.setRegion("EU");

 // RightFeeParameter class
var myrightsfeeparameters_2 = new RightsFeeParameters();
myrightsfeeparameters_2.setFee(1.23);
myrightsfeeparameters_2.setFeeType(1);
myrightsfeeparameters_2.setCurrency("EURO");
myrightsfeeparameters_2.setBankAccount("Deusche Bank");
```

```
    // Parameters for method addGrantForEndUser

// rightsparameters class
var myrightsparameters_3 = new RightsParameters();
myrightsparameters_3.setDistGrantId("dist grant id");
myrightsparameters_3.setRight("Right");
myrightsparameters_3.setAdaptationRules("adapatation rules");

    // RightsIntervalParameters
var myrightsintervalParameters_3= new RightsIntervalParameters();
myrightsintervalParameters_3.setValidityInterval (true);
myrightsintervalParameters_3.setNotBefore("nbefore");
myrightsintervalParameters_3.setNotAfter ("nafter");

    //RightsLimitParameters
var myrightslimitparameters_3  = new RightsLimitParameters();
myrightslimitparameters_3.setCountLimit (true);
myrightslimitparameters_3.setLimit (1);

    // RightsRegionParameters
var myrightsregionparameters_3 = new RightsRegionParameters();
myrightsregionparameters_3.setValidityRegion (true);
myrightsregionparameters_3.setCountry("Germany");
myrightsregionparameters_3.setRegion("EU");



    // RightFeeParameter class
var myrightsfeeparameters_3 = new RightsFeeParameters();
myrightsfeeparameters_3.setFeeType(1);
myrightsfeeparameters_3.setFee(1.23);
myrightsfeeparameters_3.setCurrency("EURO");
myrightsfeeparameters_3.setBankAccount("Deusche Bank");



// Parameters for method addPARGrantEndUser
// PARGrantParameters class
var mypargrantparameters_1 = new PARGrantParameters();
mypargrantparameters_1.setPARTmpId("PAR Tmp ID");

// rightsparameters class
var myrightsparameters_4 = new RightsParameters();
myrightsparameters_4.setAXOID("AXOID");
myrightsparameters_4.setDiType(1);
myrightsparameters_4.setDiSubType(1);
myrightsparameters_4.setRight("Right");
myrightsparameters_4.setAdaptationRules ("adp rules");
    // RightsIntervalParameters
var myrightsintervalParameters_4= new RightsIntervalParameters();
myrightsintervalParameters_4.setValidityInterval (true);
myrightsintervalParameters_4.setNotBefore("nbefore");
myrightsintervalParameters_4.setNotAfter ("nafter");

    //RightsLimitParameters
var myrightslimitparameters_4  = new RightsLimitParameters();
myrightslimitparameters_4.setCountLimit (true);
myrightslimitparameters_4.setLimit (1);

    // RightsRegionParameters
var myrightsregionparameters_4 = new RightsRegionParameters();
myrightsregionparameters_4.setValidityRegion (true);
myrightsregionparameters_4.setCountry("Germany");
myrightsregionparameters_4.setRegion("EU");



    // RightFeeParameter class
var myrightsfeeparameters_4 = new RightsFeeParameters();
myrightsfeeparameters_4.setFeeType(1);
myrightsfeeparameters_4.setFee(1.23);
myrightsfeeparameters_4.setCurrency("EURO");
myrightsfeeparameters_4.setBankAccount("Deusche Bank");
```

```
// Parameters for method addPARGrantForDistributor


    // PARGrantParameters class
var mypargrantparameters_2 = new PARGrantParameters();
mypargrantparameters_2.setPARTmpId("PAR Tmp ID");

    // RightsParameters class
var myrightsparameters_5 = new RightsParameters();
myrightsparameters_5.setAXOID("AXOID");
myrightsparameters_5.setDiType(1);
myrightsparameters_5.setDiSubType(1);
    // RightsIntervalParameters
var myrightsintervalParameters_5 = new RightsIntervalParameters();
myrightsintervalParameters_5.setValidityInterval (true);
myrightsintervalParameters_5.setNotBefore("nbefore");
myrightsintervalParameters_5.setNotAfter ("nafter");

    //RightsLimitParameters
var myrightslimitparameters_5  = new RightsLimitParameters();
myrightslimitparameters_5.setCountLimit (true);
myrightslimitparameters_5.setLimit (1);

    // RightsRegionParameters
var myrightsregionparameters_5 = new RightsRegionParameters();
myrightsregionparameters_5.setValidityRegion (true);
myrightsregionparameters_5.setCountry("Germany");
myrightsregionparameters_5.setRegion("EU");

    // RightFeeParameter class
var myrightsfeeparameters_5 = new RightsFeeParameters();
myrightsfeeparameters_5.setFeeType(1);
myrightsfeeparameters_5.setFee(1.23);
myrightsfeeparameters_5.setCurrency("EURO");
myrightsfeeparameters_5.setBankAccount("Deusche Bank");



// Parameters for method addPARGrantForEndUser


    // PARGrantParameters class
var mypargrantparameters_3 = new PARGrantParameters();
mypargrantparameters_3.setPARTmpId("PAR Tmp ID");

    // RightsParameters class
var myrightsparameters_6 = new RightsParameters();
myrightsparameters_6.setDistGrantId("distination grant id");
myrightsparameters_6.setRight("Right");
myrightsparameters_6.setAdaptationRules("adaptation rules");

    // RightsIntervalParameters
var myrightsintervalParameters_6 = new RightsIntervalParameters();
myrightsintervalParameters_6.setValidityInterval (true);
myrightsintervalParameters_6.setNotBefore("nbefore");
myrightsintervalParameters_6.setNotAfter ("nafter");

    //RightsLimitParameters
var myrightslimitparameters_6  = new RightsLimitParameters();
myrightslimitparameters_6.setCountLimit (true);
myrightslimitparameters_6.setLimit (1);

    // RightsRegionParameters
var myrightsregionparameters_6 = new RightsRegionParameters();
myrightsregionparameters_6.setValidityRegion (true);
myrightsregionparameters_6.setCountry("Germany");
myrightsregionparameters_6.setRegion("EU");

    // RightFeeParameter class
var myrightsfeeparameters_6 = new RightsFeeParameters();
myrightsfeeparameters_6.setFeeType(1);
myrightsfeeparameters_6.setFee(1.23);
myrightsfeeparameters_6.setCurrency("EURO");
myrightsfeeparameters_6.setBankAccount("Deusche Bank");
```

```
// Parameters for method authorise
     // AXIDParameters class

var myaxidparameters_1 = new AXIDParameters();
myaxidparameters_1.setAXCID("AXCID");
myaxidparameters_1.setAXDID("AXDID");
myaxidparameters_1.setAXOID("AXOID");
myaxidparameters_1.setAXTID("AXTID");
myaxidparameters_1.setAXUID("1");
myaxidparameters_1.setAXWID("AXWID");

    // OpInfoParameters class
var myopinfoparameters_1 = new OpInfoParameters();
myopinfoparameters_1.setEstimatedHwFingerPrint("estimated hw finger print");
myopinfoparameters_1.setExecutionTimeStamp("Execution Time");
myopinfoparameters_1.setOperationDetailsID("operations details id");
myopinfoparameters_1.setOperationID ("operation Id");
myopinfoparameters_1.setOwnerName("owner Name");
    // ProtInfoParameters

var myprotinfoparameters_1 = new ProtInfoParameters();
myprotinfoparameters_1.setVersion("object version");
myprotinfoparameters_1.setProtStamp("proteciton stamp");
myprotinfoparameters_1.setProtInfoReq(false);

// parameter for method verify
    //byteArray class
var mybytearray_1 = new byteArray();

// parameter for method updateProtectionInfoREMOTE


// ProtInfoParameters

var myprotinfoparameters_2 = new ProtInfoParameters();
myprotinfoparameters_2.setVersion("object version");
myprotinfoparameters_2.setProtStamp("proteciton stamp");
myprotinfoparameters_2.setProtInfo("protection info required");

// parameters for the method certify

var myaxtoolinfoparameters_1 = new AXToolInfoParameters();
myaxtoolinfoparameters_1.setAxtid("Axtid");
myaxtoolinfoparameters_1.setAxrtid("Axrtid");
myaxtoolinfoparameters_1.setToolFingerPrint("toolfingerprint");
myaxtoolinfoparameters_1.setRegDeadLine("registration deadline");




rval= "\n PMSClient 1  \n\t initlicense: "+ mypmsclient_1.initLicenseEndUser( "IssuerAXUID")+
                                    mypmsclient_1.addGrantEndUser( myrightsparameters_1,
myrightsintervalParameters_1,myrightslimitparameters_1,
                                    myrightsregionparameters_1,myrightsfeeparameters_1)+
                                    mypmsclient_1.finaliseLicenseEndUser("licenseTmpId")+
                                    mypmsclient_1.initLicenseDistributor("IssuerAXUID")+

mypmsclient_1.addGrantforDistributor(myrightsparameters_2,myrightsintervalParameters_2,

myrightslimitparameters_2,myrightsregionparameters_2,myrightsfeeparameters_2)+

mypmsclient_1.addGrantforEndUser(myrightsparameters_3,myrightsintervalParameters_3

,myrightslimitparameters_3,myrightsregionparameters_3,myrightsfeeparameters_3)+
                                    mypmsclient_1.finaliseLicenseDistributor("licenseTmpId")+
                                    mypmsclient_1.initPAREndUser()+

mypmsclient_1.addPARGrantEndUser(mypargrantparameters_1,myrightsparameters_4,myrightsintervalPara
meters_4,

myrightslimitparameters_4,myrightsregionparameters_4,myrightsfeeparameters_4)+
                                    mypmsclient_1.finalisePAREndUser("PARTmpId")+
                                    mypmsclient_1.initPARDistributor()+
```

```
mypmsclient_1.addPARGrantforDistributor(mypargrantparameters_2,myrightsparameters_5,myrightsinter
valParameters_5,
                                        myrightslimitparameters_5
,myrightsregionparameters_5,myrightsfeeparameters_5)+

mypmsclient_1.addPARGrantforEndUser(mypargrantparameters_3,myrightsparameters_6,myrightsintervalP
arameters_6,
                                        myrightslimitparameters_6
,myrightsregionparameters_6,myrightsfeeparameters_6)+
                                        mypmsclient_1.finalisePARDistributor("licenseTmpId")+
                                        mypmsclient_1.getLicense("licenseId")+
                                        mypmsclient_1.sendLicense("XMLLicense")+
                                        mypmsclient_1.getPAR("string PARId")+
                                        mypmsclient_1.sendPAR("XMLPAR")+

mypmsclient_1.authorise(myaxidparameters_1,myopinfoparameters_1,myprotinfoparameters_1)+
"\n PMSClient 1  \n\t Certification Results:"+
mypmsclient_1.certify(myaxtoolinfoparameters_1,"axdom")+
"\n PMSClient 1  \n\t Verification Results:"+
mypmsclient_1.verify("axid","axrtid",mybytearray_1)+
"\n PMSClient 1  \n\t Reverification Results:"+
mypmsclient_1.reverify("axid","axrtid","toolfingerprint")+
"\n PMSClient 1  \n\t Verification Results:"+ mypmsclient_1.verifyUser("axid","axdom")+
 "\n PMSClient 1  \n\t getprotectinfolocal:"+ mypmsclient_1.getProtectionInfoLOCAL ("object",
"version", "protstamp")+
 "\n PMSClient 1  \n\t updateProtectinfoRemote:"+ mypmsclient_1.updateProtectionInfoRemote
("id",myprotinfoparameters_2 )+
 "\n PMSClient 1  \n\t isAlive:"+mypmsclient_1.isAlive()+
 "\n PMSClient 1  \n\t generateTranslation:"+ mypmsclient_1.generateTranslation
("license","originalRel","destinationRel" )+
 "\n PMSClient 1  \n\t verifyTempDistLicenseAgainstPAR:"+
mypmsclient_1.verifyTempDistLicenseAgainstPAR ("distrLicense", "par")+
 "\n PMSClient 1  \n\t verifyTempDistLicenseAgainstPARDataBase:"+
mypmsclient_1.verifyTempDistLicenseAgainstPARDatabase ("distrLicense",
                                                                    "pardatabase",
"pardatabasehost", "pardatabaseuser","pardatabasepass"),
 "\n PMSClient 1  \n\t verifyLicense:"+mypmsclient_1.verifyLicense ("license")+
  "\n PMSClient 1  \n\t verifyPAR:"+ mypmsclient_1.verifyPAR ("par") ;
```

11.10.5 Integration and compilation issues

Please refer to section "AXMEDIS DATA Types and Functions for JavaScript".for information about integration an compilation issues of JS modules within the AXCP Rule Editor and Engine.

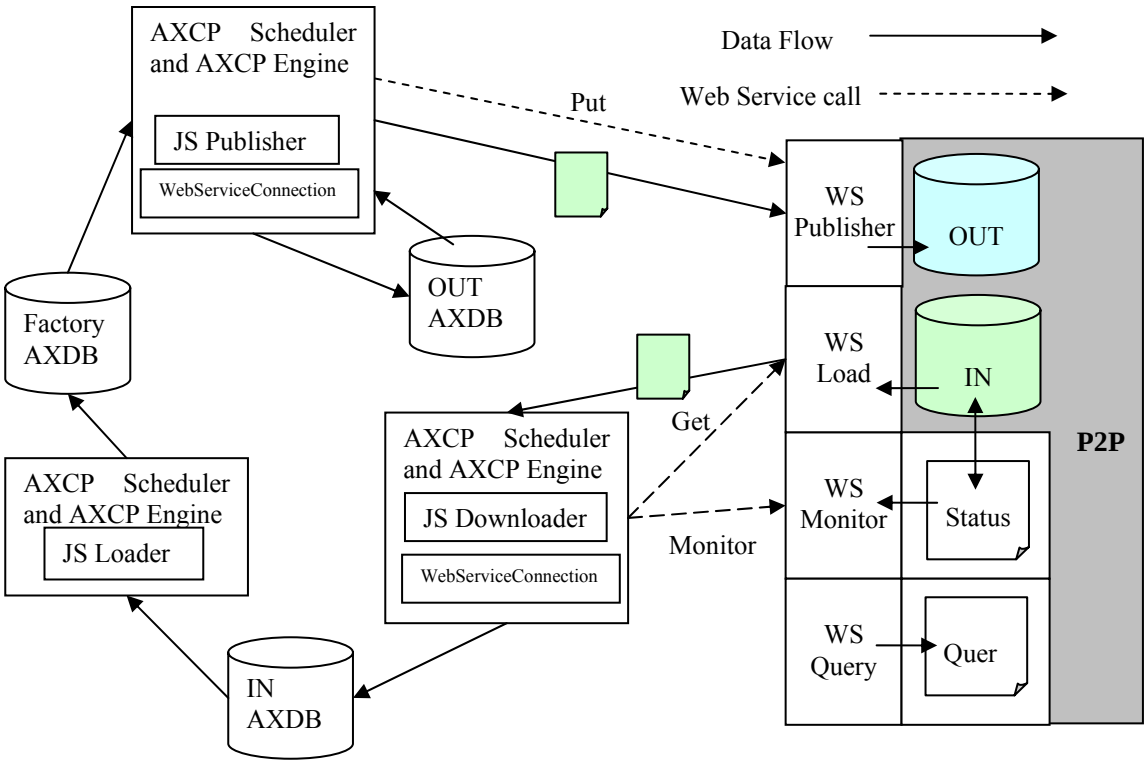
11.11 JS Classes for the P2P functioning in B2B (DSI)

Module Profile	
JSP2P Classes	
Responsible Name	Ivan Bruno
Responsible Partner	DSI
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	
Executable or Library/module (Support)	Module
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/ https://cvs.axmedis.org/repos/Framework/source/jsaxclasses/
Reference to the AXFW	https://cvs.

location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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
Wsdlpull	Wsdlpull 1.11	LGPL Licenze
LibCurl	Latest stable	LGPL license
wxWidget	Latest stable	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL Licence

The following structure have been identified to delegate the movements of objects to instances of the AXCP Engine, thus simplifying the work related to the creation of the P2P. This approach allows to decouple the database of the AXMEDIS factory with respect to the databases of input and output of the P2P B2B network. Continuos line represent the flow of the Data amongs Databases and P2P environment, dashed line explains the functional link amongs Databases and P2P environment



All the databases are instance of the AXDB. The AXCP Scheduler and Engine are instances of the AXCP tools. All the JS modules are used for creating scripts and automating the movements of objects from and to the P2P B2B Network represented on the right as P2P. The In and Out DB are the possible internal Databases or filesytem provided by the P2P tool. WS stands for WebServices provided by the interface that allows using the internal functionalities of the P2P tool.
JS Publisher, JSLoader and JSDownloader are the Javascript classes that will expose methods to realize the movement of the object, in particular the JSPublisher and JSDownloader could host an instance of the *WebServiceConnection* class to realize the internal client for directly accessing to WebServices. Alternatively, the WebServiceConnection class could be used in the script by using the JSWebConnetion object and realize the WebService access indirectly via script.

11.11.1 Module Design in terms of Classes

JS P2PClient classes are a set of Javascript classes that provide the Web Service support to the AXCP Engine by modeling the Web Service client that interacts with the AXEPTool. They consist in:

- **JSAXP2PManager** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol

- **JSDownloadedObjectListResponse** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSDownloadStatusResponse** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSPublicationStatusResponse** The class models a meta class for managing web services by loading WSDL description and dynamically creating services (methods).
- **JSPublishedObjectListResponse** The class models a simple form of a http web page
- **JSPublicationStatusResponse** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
- **JSSystemInformationsResponse** The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol

11.11.2 Technical and Installation information

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section "AXMEDIS DATA Types and Functions for JavaScript".

References to other major components needed	P2P
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

11.11.3 Draft User Manual

This is a draft User Manual for the JS P2P classes.

JSP2PClient

Class name in Javascript is *AXP2PManager*

Exposed Properties

string uri
The AXEPTool's Uri.

string trackeruri
The Tracker's Uri.

string lastError
Last gsoap error.

int timeout
Gsoap call's timeout in seconds.

Exposed methods

string getUri()
Gets the endpointUri of the AXEPTool.

boolean setUri(string Uri)
Sets the endpointUri of the AXEPTool.

boolean setTrackerUri(string Uri)
Sets the TrackerUri.

string getLastError()
Gets the last gsoap error.

boolean setTimeout(int timeout)
Sets the gsoap call's timeout in seconds.

array getPublicationStatus(string Uri)
Provides the current status of a given object which is in the Published Object Area of the P2P enviroment and selected by the proper identification (AXOID or local file name). It returns the amount of bytes, seeds and peers for each segment, and the list of contacting peers for a given media.

boolean removePublishedObject(string Uri)
Deletes the object specified by the Uri (AXOID or local file name) from the database/directory of Published objects. This is something that should not be done in P2P systems, but when they are used for B2B the directory could reach the limit of their size and housekeeping is needed.

string getDownloadedObjectURL(string Uri)
Retrieves the object URI (the object is given by its AXOID or local file name) from which the downloaded at 100% object can be copied into the AXMEDIS database by the AXCP. This action wil be performed by the AXCP directly accessing to that URL.

boolean removeDownloadedObject(string Uri)
Deletes the object specified by the Uri (AXOID or local file name) from the directory of Downloaded objects. This is something that should not be done in P2P systems, but in some cases the directory could reach the limit of their size and housekeeping is needed.

boolean downloadObject(string Uri)
Starts the download of the object specified by the AXOID in the P2P enviroment. The AXOID field has to be added into the Tracker database. It could be used with traditional media if torrentifile are at disposal.

boolean downloadObjectURL(string Uri)
Starts the download of the object specified by the bittorrent URL in the P2P enviroment. It could be used with traditional media if torrentifile are at disposal.

boolean controlDownloadingObject(string Uri, boolean start)
Start/Stop the download of the object specified (given by its Uri, which can be its AXOID or local file name).

array getDownloadStatus(string Uri)
Provides the current status of a downloading object (given by its Uri, which can be its AXOID or local file name) in the P2P enviroment. It returns the current percentage of download, estimated time to completion (if possible), peers and seeds.

boolean publishObject(string Uri)
Published the object (given by its filepath) in the Published Object repository of the P2P client tool and create a bittorrent file, thus posting bittorrent file on the selected tracker URL.

array getPublishedObjectList()
Returns a list of objects which are present in the directory of Published objects. The list depicts all media (both AXMEDIS or general), including file name, AXOID (if any) and infoHash.

array getDownloadingObjectList()
Returns the list of objects in the directory of Downloaded objects.

string getVersion()
Gets the AXEPTool's version.

string getStatus()
Gets an object's status.

11.11.4 Examples of usage

```
var start = true;
var axeptoolUri = "http://localhost:7780/WebServices/P2PMonitoring";
var trackerUri = "http://axtrk.axmedis.org:8080/AXTrackv2/";
var axmedisObjUri = "URN:AXMEDIS:00000:OBJ:C6921986-7E79-3753-863B-D6870D143510";

var manager = new AXP2PManager(axeptoolUri, trackerUri);
```

JSDownloadedObjectListResponse

Class name in Javascript is *DownloadedObjectListResponse*

Exposed Properties

array downloadedObjectList
List of downloaded objects as a matrix of strings where each row provides values for *AXOID* (column 0), *InfoHash* (column 1) and *LocalFilename* (column 2).

Exposed methods

string getAXOID(int index)
Gets the AXOID of the donwloaded object referred by *index*. If index is greater than the size of the donwloaded object list the method returns an empty string.

string getinfoHash(int index)
Gets the infoHash of the donwloaded object referred by *index*. If index is greater than the size of the donwloaded object list the method returns an empty string.

string getlocalFileName(int index)
Gets the local file name of the donwloaded object referred by *index*. If index is greater than the size of the donwloaded object list the method returns an empty string.

array getDownloadedObjectList()
Gets the downloaded object list as a matrix of strings where each row provides values for *AXOID* (column 0), *InfoHash* (column 1) and *LocalFilename* (column 2).

11.11.5 Examples of usage

```
var start = true;
var axeptoolUri = "http://localhost:7780/WebServices/P2PMonitoring";
var trackerUri = "http://axtrk.axmedis.org:8080/AXTrackv2/";
var axmedisObjUri = "URN:AXMEDIS:00000:OBJ:C6921986-7E79-3753-863B-D6870D143510";

var manager = new AXP2PManager(axeptoolUri, trackerUri);

if(!manager.downloadObject(axmedisObjUri))
  print(manager.getLastError());
else
  print("Object downloaded");
```

JSDownloadStatusResponse

Class name in Javascript is *DownloadStatusResponse*

Exposed Properties

int nseeds
Number of seeds.

int dbytes
Amount of downloaded Bytes.

int ela
Elapsed download time.

int eta
Estimated download time.

int dperc
Downloaded percentage.

array peerslist
The list of peers.

Exposed methods

int getdperc()
Gets the downloaded percentage.

int getdBytes()
Gets the downloaded Bytes.

int getela()
Gets the download elapsed time.

int geteta()
Gets the estimated download time.

int getnseeds()
Gets the number of seeds.

array getpeerslist()
Gets the peers list.

11.11.6 Examples of usage

```
var start = true;
var axeptoolUri = "http://localhost:7780/WebServices/P2PMonitoring";
var trackerUri = "http://axtrk.axmedis.org:8080/AXTrackv2/";
var Uri = "file.axm";

var manager = new AXP2PManager(axeptoolUri, trackerUri);

var res = manager.getDownloadStatus(Uri);
```

JSPublicationStatusResponse

Class name in Javascript is *PublicationStatusResponse*

Exposed Properties

int numberOfSeeds
The number of seeds.

int uploadedBytes
The amount of uploaded Bytes.

array PeersList
The peers list.

Exposed methods

int getnseeds()
Gets the number of seeds.

int getbytes()
Gets the amount of uploaded Bytes.

array getPeersList()
Gets the peers list.

11.11.7 Examples of usage

```
var start = true;
var axeptoolUri = "http://localhost:7780/WebServices/P2PMonitoring";
var trackerUri = "http://axtrk.axmedis.org:8080/AXTrackv2/";
var Uri = "file.axm";

var manager = new AXP2PManager(axeptoolUri, trackerUri);

var res = manager.getPublicationStatus(Uri);
```

JSSystemInformationsResponse

Class name in Javascript is *SystemInformationsResponse*

Exposed Properties

string configFile
The AXEPTool's configuration file.

int cpu
The AXEPTool machine's cpu type.

int memory
The AXEPTool machine's amount of free RAM.

string os
The AXEPTool machine's Operating System.

string username
The AXEPTool machine's user name.

string version
The AXEPTool's version.

Exposed methods

string getConfigFile()
Gets the AXEPTool's configuration file.

int getCpu()
AXEPTool machine's cpu type.

int getMemory()
Gets AXEPTool machine's amount of free RAM.

string getOS()
Gets the AXEPTool machine's Operating System.

string getUserName()
Gets the AXEPTool machine's user name.

string getVersion()
Gets the AXEPTool's version.

11.11.8 Examples of usage

```
var start = true;
var axeptoolUri = "http://localhost:7780/WebServices/P2PMonitoring";
var trackerUri = "http://axtrk.axmedis.org:8080/AXTrackv2/";
var Uri = "file.axm";

var manager = new AXP2PManager(axeptoolUri, trackerUri);

var res = manager.getSystemInformations();
```

11.12 JSFunctions (DSI)

Module Profile		
JSFunction Classes		
Responsible Name	Ivan Bruno	
Responsible Partner	DSI	
Status (proposed/approved)	approved	
Implemented/not implemented	Implemented	
Status of the implementation	50%	
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows, Linux, Mac OS X	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/functions/ https://cvs.axmedis.org/repos/Framework/source/jsaxclasses/functions/	
Reference to the AXFW location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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
wxWidget	*wxWidget 2.4.2*	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL Licence

JS Functions is a set of auxiliary functions wrappred into Javascript. They will be divided into the following categories:

Statistical

- MAX – return the maximum value of an numerical array
- MIN - return the minimum value of an numerical array
- VAR – return the variance of an numerical array
- AVERAGE – return the average of an numerical array
- MODE – return the mode of an numerical array

Combinatorial

- Data Permutation
- Sort data

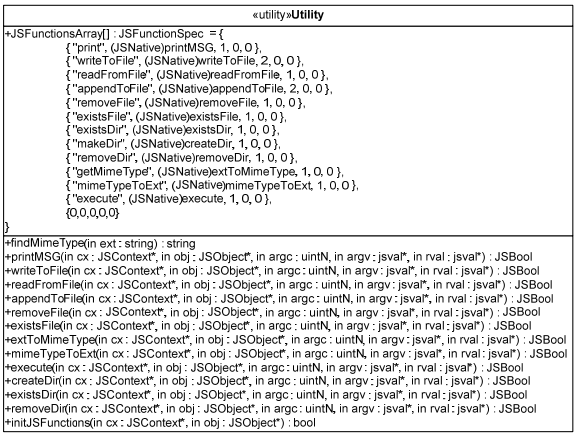
Set Management

- *Intersection* – $A \cap B$, it will return the list of common items
- *Union* – $A \cup B$, it will return the list of items as union of sets
- *Inclusion* – return true if $A \subseteq B \neq \emptyset$

Generic

- File system functions – File exists, dir exists, create dir, etc…
- Communication functions – A function will provide the support for the communication via JavaScript. Messages will be routed via the Engine to the Axmedis Workflow Manager.

11.12.1 Module Design in terms of Classes



11.12.2 Draft User Manual

This set of JavaScript functions wraps utilities listed below:

I/O functions - for managing the I/O via script

File functions - for managing files

Dir functions - for managing folders

Process Functions - for managing external process and applications

Mime Type Functions - for converting extension into mime type and viceversa

Network functions - for managing network utilities

I/O functions

print(string msg)
Print a string on the current I/O device. It could be the GUI, the console or other.

alert(string msg)
As "Print" displays a string on the current I/O device. It could be the GUI, the console or other.

File functions

boolean writeToFile(string filePath, string buffer)
Write a string buffer to a file at filePath

boolean appendToFile(string filePath, string buffer)
Append a string buffer to a file at filePath

boolean removeFile(string filePath)
Remove a file

boolean existsFile(string filePath)
Test if a file at filePath exists

string readFromFile(string filePath)
Return a string buffer from a file at filePath

boolean copyFile(string filePath,string targetPath)
Copy the file at filePath to the targetPath returning TRUE in the success case.

boolean renameFile(string filePath,string extension)
Command to change the extension of a file at filePath returning TRUE in the success case.

string getFirstFile(string path,string wildcard)
Provide the first file in the path folder. The wildcard parameter is a filter on files extension ("*.*", "*.txt", etc.). The default value for wildcard parameter is "*.*"If the directory has not files null is returned

string getNextFile(string path,string wildcard)
Provide the first file in the path folder. The wildcard parameter is a filter on files extension ("*.*", "*.txt", etc.). The default value for wildcard parameter is "*.*". If the directory has not files null is returned. This function has to be used in conjuction with the getFirstFile.

array getAllFiles(string path, string wildcard)
Search all files matching the wildcard in the path dir and in all subdir recursively.

number getFileLength(string fullPath)
It returns the size of file in Bytes. The function returns -1 if the file does not exist.

Dir functions

boolean makeDir(string dirPath)
Create a directory at dirPath returning TRUE in the success case.

boolean removeDir(string dirPath)
Remove the directory at dirPath returning TRUE in the success case.

boolean existsDir(string dirPath)
Test if the directory at dirPath exists

array listDir(string path,string wildcard)
Provide the list of directories in the path folder. The wildcard parameter is a filter on the extension of directories ("*.*", "*.txt", etc.). The default value for wildcard parameter is "*.*"

Process Functions

boolean execute(string commandLine)
Invoke the execution of an external process/application by means the command line and parameters. The call is synchronous and returns TRUE in the success case.

Mime Type Functions

string getMimeType(string ext)
Get the mime type from the file extension

string mimeTypeToExt(string mimetype)
Get the extension from the mime type

Network Functions

string getIPAddress()
Get the IP Address of the machine as string

string getHostName(string mimetype)
Get the name of the host

11.12.3 Examples of usage

The script parse an xml string loaded from filesystem the xml is loaded by means the 'readFromFile' function that returns a string with the xml. The ShowXML calls the function in the showXML script. It uses the XML parser provided by the "rexml" script. Such script define an XML object in JS and gives some tools to browse and retrieve information from the xml elements

```
var str = readFromFile(xml);
var obj = new AxMetadata();
obj.addXML(str);
var xml = obj.getXML();
ShowXML(xml);

function ShowXML(strXML) {
      var xmlDoc = new REXML(strXML);
      print("The root element " + xmlDoc.rootElement.name + " has " +
      xmlDoc.rootElement.childElements.length + " child elements.");
      for (var i=0; i<xmlDoc.rootElement.childElements.length; i++) {
            var item = xmlDoc.rootElement.childElements[i];
            print("Child element of type " + item.type + " "+item.name );
        }
}
```

11.12.4 Technical and Installation information

The JSFunctions comes in the form of a static LIB. The initJSFunction has to be called in the source code where the Engine is initialized. This is necessary to add the set of functions to the Javascript Engine.

11.13 JSUserProfile (UR)

Module Profile	
JSUserProfile	
Responsible Name	Badii

Responsible Partner	UR (previously IRC)	
Status (proposed/approved)	approved	
Implemented/not implemented	Implemented	
Status of the implementation	100%	
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows, Linux, Mac OS X	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/profiling/ https://cvs.axmedis.org/repos/Framework/source /jsaxclasses/profiling/	
Reference to the AXFW location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section

Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
	Java Script (wrappers)	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces	Xerces 2.7.0	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL license

The user profile contains information related to the users, the usage and the user preferences. This profile should be managed at the personalization server and be able to store minimal details about the user to identify his personal preferences in respect to the services being offered. The elements in this profile are the minimal set of elements that is required to meet the AXMEDIS User's personalization needs. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-schema.xsd. The JSUserProfile is a JS class to automatically create and manage user profiles as a part of rule scripts for AXCP engine.

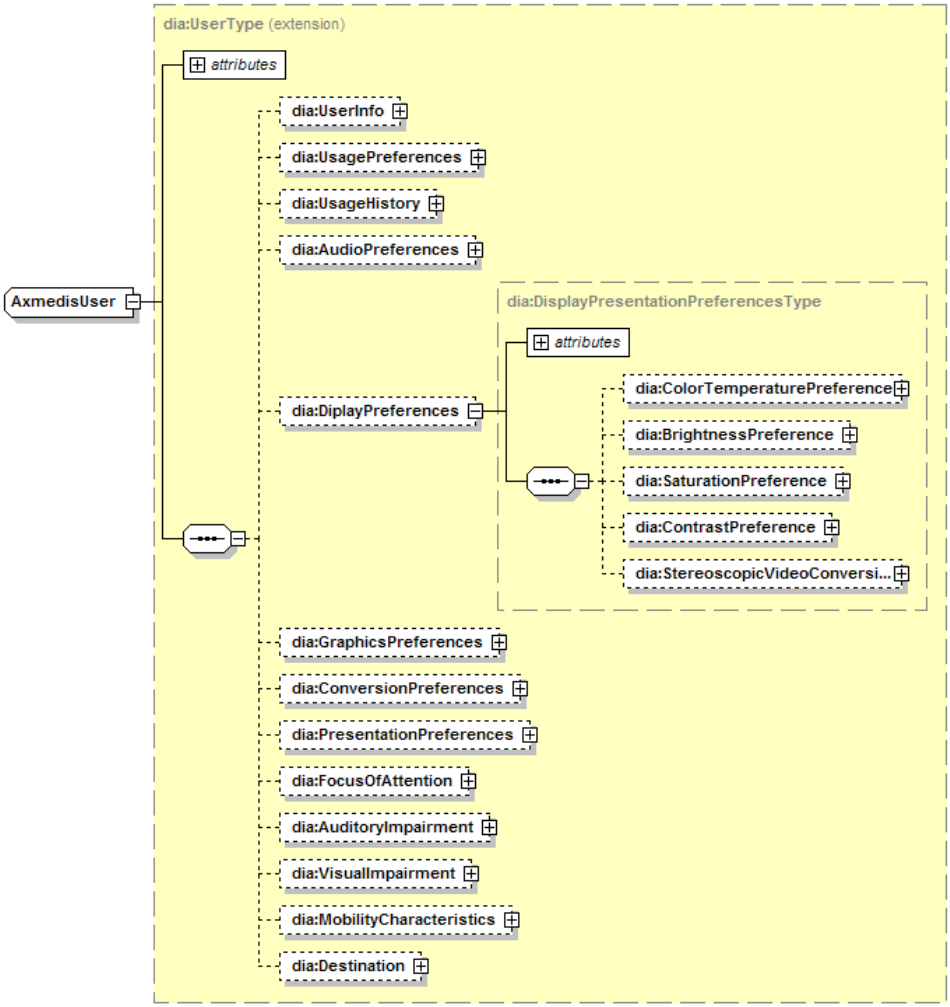
The functionalities and values for the user profiles can be accessed through the methods given below. These functions are only wrappers for the functions in the c++ class UserProfile.

- *UserProfile:* Constructor for Initialising the Profile
- *loadXmlFile:* Loading the Xml from file
- *loadXmlString:* Loading Xml from an Xml String
- *createXmlfile:*Create and Xml file where the file is specified as uri
- *getXml:* Returns the contents of the XML file in String form
- *setXml:* Write the string as the contents of the XML file
- *getAttribute:* It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setAttribute:* Sets an individual values to the attributes of the xml file
- *getValue:* It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setValue:* Sets an individual values to the elements of the xml file

- *createElement:* Create an xml element using the DomPath and Element Name
- *deleteElement:* Delete an xml element based on the DomPath
- *close*: Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.13.1 Module Design in terms of Classes

The user profile is a valid XML file based on following Schema. For entire schema file please refer to the Axmedis-schema.xsd.




```
<?xml version="1.0"?>
<!-- Digital Item Adaptation ISO/IEC 21000-7 -->
<!-- Schema for Usage Environment Description Tools -->
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" targetNamespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="ISO/IEC 21000-7" id="axmedis-UED.xsd">
  <import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg7-udp-2003.xsd"/>
  <include schemaLocation="DIA.xsd"/>
  <!-- ################################################   -->
  <!--  Definition of UsageEnvironment                    -->
  <!-- ################################################   -->
```

```
<complexType name="UsageEnvironmentType">
  <complexContent>
    <extension base="dia:DIADescriptionType">
        <sequence>
            <element name="UsageEnvironmentProperty" type="dia:UsageEnvironmentPropertyBaseType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="UsageEnvironmentPropertyBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIADescriptionType"/>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of Users        -->
<!-- ################################################   -->
<complexType name="UsersType">
  <complexContent>
    <extension base="dia:UsageEnvironmentPropertyBaseType">
        <sequence>
            <element name="User" type="dia:UserType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of User   -->
<!-- ################################################   -->
<complexType name="UserType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="UserInfo" type="dia:UserInfoType" minOccurs="0"/>
            <element name="UsagePreferences" type="dia:UsagePreferencesType" minOccurs="0"/>
            <element name="UsageHistory" type="dia:UsageHistoryType" minOccurs="0"/>
            <element name="AudioPreferences" type="dia:AudioPresentationPreferencesType" minOccurs="0"/>
            <element name="DiplayPreferences" type="dia:DisplayPresentationPreferencesType" minOccurs="0"/>
            <element name="GraphicsPreferences" type="dia:GraphicsPresentationPreferencesType" minOccurs="0"/>
            <element name="ConversionPreferences" type="dia:ConversionPreferenceType" minOccurs="0"/>
            <element name="PresentationPreferences" type="dia:PresentationPriorityPreferenceType" minOccurs="0"/>
            <element name="FocusOfAttention" type="dia:FocusOfAttentionType" minOccurs="0"/>
            <element name="AuditoryImpairment" type="dia:AuditoryImpairmentType" minOccurs="0"/>
            <element name="VisualImpairment" type="dia:VisualImpairmentType" minOccurs="0"/>
            <element name="MobilityCharacteristics" type="dia:MobilityCharacteristicsType" minOccurs="0"/>
            <element name="Destination" type="dia:DestinationType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="UserCharacteristicBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIABaseType"/>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of UserInfo        -->
<!-- ################################################   -->
<complexType name="UserInfoType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="UserInfo" type="mpeg7:AgentType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
```

```
</complexType>
<!-- ################################################   -->
<!--  Definition of UsagePreferences     -->
<!-- ################################################   -->
<complexType name="UsagePreferencesType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="UsagePreferences" type="mpeg7:UserPreferencesType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of UsageHistory            -->
<!-- ################################################   -->
<complexType name="UsageHistoryType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="UsageHistory" type="mpeg7:UsageHistoryType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################ -->
<!--  Definition of AudioPresentationPreferences              -->
<!-- ################################################ -->
<complexType name="AudioPresentationPreferencesType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="VolumeControl" type="mpeg7:zeroToOneType" minOccurs="0"/>
            <element name="FrequencyEqualizer" type="dia:FrequencyEqualizerType" minOccurs="0"/>
            <element name="AudibleFrequencyRange" minOccurs="0">
                <complexType>
                  <sequence>
                      <element name="StartFrequency" type="float"/>
                      <element name="EndFrequency" type="float"/>
                  </sequence>
                </complexType>
            </element>
            <element name="AudioOutputDevice" minOccurs="0">
                <simpleType>
                  <restriction base="string">
                      <enumeration value="Headphone"/>
                      <enumeration value="Loudspeaker"/>
                  </restriction>
                </simpleType>
            </element>
            <element name="BalancePreference" minOccurs="0">
                <simpleType>
                  <restriction base="float">
                      <minInclusive value="-10"/>
                      <maxInclusive value="10"/>
                  </restriction>
                </simpleType>
            </element>
            <element name="Soundfield" type="dia:SoundfieldType" minOccurs="0"/>
            <element name="SoniferousSpeed" minOccurs="0">
                <simpleType>
                  <restriction base="float">
                      <minExclusive value="0"/>
                  </restriction>
                </simpleType>
            </element>
        </sequence>
```

```
      </extension>
    </complexContent>
  </complexType>
  <simpleType name="FrequencyEqualizerType">
    <restriction>
      <simpleType>
          <list itemType="dia:FrequencyEqualizerBaseType"/>
      </simpleType>
      <length value="31"/>
    </restriction>
  </simpleType>
  <simpleType name="FrequencyEqualizerBaseType">
    <restriction base="float">
      <minInclusive value="-15"/>
      <maxInclusive value="15"/>
    </restriction>
  </simpleType>
  <complexType name="SoundfieldType">
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
              <element name="ImpulseResponse" type="dia:ImpulseResponseType" minOccurs="0"/>
              <element name="PerceptualParameters" type="dia:PerceptualParametersType" minOccurs="0"/>
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="ImpulseResponseType">
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
              <element name="SamplingFrequency" type="mpeg7:nonNegativeReal" minOccurs="0"/>
              <element name="BitsPerSample" type="nonNegativeInteger" minOccurs="0"/>
              <element name="NumOfChannels" type="nonNegativeInteger" minOccurs="0"/>
          </sequence>
          <attribute name="href" type="anyURI" use="optional"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="PerceptualParametersType">
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
              <element name="SourcePresence" type="float" minOccurs="0"/>
              <element name="SourceWarmth" type="float" minOccurs="0"/>
              <element name="SourceBrilliance" type="float" minOccurs="0"/>
              <element name="RoomPresence" type="float" minOccurs="0"/>
              <element name="RunningReverberance" type="float" minOccurs="0"/>
              <element name="Envelopment" type="float" minOccurs="0"/>
              <element name="LateReverberance" type="float" minOccurs="0"/>
              <element name="Heavyness" type="float" minOccurs="0"/>
              <element name="Liveness" type="float" minOccurs="0"/>
              <element name="RefDistance" type="float" minOccurs="0"/>
              <element name="FreqLow" type="float" minOccurs="0"/>
              <element name="FreqHigh" type="float" minOccurs="0"/>
              <element name="TimeLimit1" type="float" minOccurs="0"/>
              <element name="TimeLimit2" type="float" minOccurs="0"/>
              <element name="TimeLimit3" type="float" minOccurs="0"/>
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of DisplayPresentationPreferences          -->
  <!-- ################################################   -->
  <complexType name="DisplayPresentationPreferencesType">
    <complexContent>
```

```
      <extension base="dia:UserCharacteristicBaseType">
        <sequence>
          <element name="ColorTemperaturePreference" type="dia:ColorPreferenceType" minOccurs="0"/>
          <element name="BrightnessPreference" type="dia:ColorPreferenceType" minOccurs="0"/>
          <element name="SaturationPreference" type="dia:ColorPreferenceType" minOccurs="0"/>
          <element name="ContrastPreference" type="dia:ColorPreferenceType" minOccurs="0"/>
          <element name="StereoscopicVideoConversion" type="dia:StereoscopicVideoConversionType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of ColorPreference       -->
  <!-- ################################################   -->
  <complexType name="ColorPreferenceType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="BinNumber" type="mpeg7:unsigned12"/>
          <element name="Value" minOccurs="0" maxOccurs="unbounded">
            <complexType>
              <sequence>
                <element name="PreferredValue" type="mpeg7:unsigned12"/>
                <element name="ReferenceValue" type="mpeg7:unsigned12"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of StereoscopicVideoConversion           -->
  <!-- ################################################   -->
  <complexType name="StereoscopicVideoConversionType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="From2DTo3DStereoscopic" minOccurs="0">
            <complexType>
              <sequence>
                <element name="ParallaxType">
                  <simpleType>
                    <restriction base="string">
                      <enumeration value="Positive"/>
                      <enumeration value="Negative"/>
                    </restriction>
                  </simpleType>
                </element>
                <element name="DepthRange" type="mpeg7:zeroToOneType"/>
                <element name="MaxDelayedFrame" type="nonNegativeInteger" minOccurs="0"/>
              </sequence>
            </complexType>
          </element>
          <element name="From3DStereoscopicTo2D" minOccurs="0">
            <complexType>
              <sequence>
                <element name="LeftRightInterVideo">
                  <simpleType>
                    <restriction base="string">
                      <enumeration value="Left"/>
                      <enumeration value="Right"/>
                      <enumeration value="Intermediate"/>
                    </restriction>
                  </simpleType>
                </element>
              </sequence>
```

```
                </complexType>
            </element>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################ -->
<!--  Definition of GraphicsPresentationPreferencesType-->
<!-- ################################################ -->
<complexType name="GraphicsPresentationPreferencesType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="GeometryEmphasis" type="mpeg7:zeroToOneType" minOccurs="0"/>
            <element name="TextureEmphasis" type="mpeg7:zeroToOneType" minOccurs="0"/>
            <element name="AnimationEmphasis" type="mpeg7:zeroToOneType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################ -->
<!--    Definition of ConversionPreference          -->
<!-- ################################################ -->
<complexType name="ConversionPreferenceType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="GeneralResourceConversions" type="dia:ResourceConversionType" minOccurs="0"/>
            <element name="SpecificResourceConversions" type="dia:SpecificResourceConversionsType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="ResourceConversionType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="Conversion" type="dia:ConversionType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="ConversionType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="From" type="mpeg7:ControlledTermUseType" minOccurs="0"/>
            <element name="To" type="mpeg7:ControlledTermUseType"/>
        </sequence>
        <attribute name="order" type="nonNegativeInteger" use="required"/>
        <attribute name="weight" type="mpeg7:nonNegativeReal" use="optional" default="1.0"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="SpecificResourceConversionsType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="Object" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                  <complexContent>
                      <extension base="dia:ResourceConversionType">
                        <attribute name="target" type="anyURI" use="required"/>
                      </extension>
                  </complexContent>
                </complexType>
            </element>
```

```
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--    Definition of PresentationPriorityPreference  -->
  <!-- ################################################ -->
  <complexType name="PresentationPriorityPreferenceType">
    <complexContent>
      <extension base="dia:UserCharacteristicBaseType">
        <sequence>
          <element name="GeneralResourcePriorities" type="dia:GeneralResourcePrioritiesType" minOccurs="0"/>
          <element name="SpecificResourcePriorities" type="dia:SpecificResourcePrioritiesType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GeneralResourcePrioritiesType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="ModalityPriorities" type="dia:ModalityPrioritiesType" minOccurs="0"/>
          <element name="GenrePriorities" type="dia:GenrePrioritiesType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="SpecificResourcePrioritiesType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="Object" minOccurs="0" maxOccurs="unbounded">
            <complexType>
              <attribute name="priorityLevel" type="mpeg7:nonNegativeReal" use="optional" default="1.0"/>
              <attribute name="target" type="anyURI" use="required"/>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="ModalityPrioritiesType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="Modality" minOccurs="0" maxOccurs="unbounded">
            <complexType>
              <complexContent>
                <extension base="mpeg7:ControlledTermUseType">
                  <attribute name="priorityLevel" type="mpeg7:nonNegativeReal" use="optional" default="1.0"/>
                </extension>
              </complexContent>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GenrePrioritiesType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="Genre" minOccurs="0" maxOccurs="unbounded">
            <complexType>
              <complexContent>
                <extension base="mpeg7:ControlledTermUseType">
                  <attribute name="priorityLevel" type="mpeg7:nonNegativeReal" use="optional" default="1.0"/>
```

```
                    </extension>
                  </complexContent>
                </complexType>
              </element>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
      <!-- ################################################ -->
      <!-- Definition of FocusOfAttentionType -->
      <!-- ################################################ -->
      <complexType name="FocusOfAttentionType">
        <complexContent>
          <extension base="dia:UserCharacteristicBaseType">
            <choice minOccurs="0">
              <element name="ROI" minOccurs="0">
                <complexType>
                  <attribute name="uri" type="anyURI" use="required"/>
                  <attribute name="updateInterval" type="mpeg7:nonNegativeReal" use="optional" default="0"/>
                </complexType>
              </element>
              <element name="TextFocusOfAttention" type="dia:TextFocusOfAttentionType" minOccurs="0"/>
              <element name="SceneObjectFocusOfAttention" type="mpeg7:MediaLocatorType" minOccurs="0"/>
            </choice>
          </extension>
        </complexContent>
      </complexType>
      <complexType name="TextFocusOfAttentionType">
        <complexContent>
          <extension base="dia:DIABaseType">
            <sequence>
              <element name="Keyword" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                  <simpleContent>
                    <extension base="mpeg7:TextualType">
                      <attribute name="preferenceValue" type="mpeg7:preferenceValueType" use="optional" default="10"/>
                    </extension>
                  </simpleContent>
                </complexType>
              </element>
              <element name="Font" type="dia:FontType" minOccurs="0"/>
            </sequence>
            <attribute name="textPresentationSpeed" type="mpeg7:nonNegativeReal" use="optional"/>
          </extension>
        </complexContent>
      </complexType>
      <complexType name="FontType">
        <complexContent>
          <extension base="dia:DIABaseType">
            <attribute name="fontColor" type="token" use="optional"/>
            <attribute name="fontSize" type="positiveInteger" use="optional"/>
            <attribute name="fontType" type="token" use="optional"/>
          </extension>
        </complexContent>
      </complexType>
      <!-- ############################################ -->
      <!-- Definition of AuditoryImpairment -->
      <!-- ############################################ -->
      <complexType name="AuditoryImpairmentType">
        <complexContent>
          <extension base="dia:UserCharacteristicBaseType">
            <sequence>
              <element name="RightEar" type="dia:AudiogramType"/>
              <element name="LeftEar" type="dia:AudiogramType"/>
            </sequence>
          </extension>
        </complexContent>
```

```
  </complexType>
  <complexType name="AudiogramType">
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
              <element name="Freq125Hz" type="float" minOccurs="0"/>
              <element name="Freq250Hz" type="float"/>
              <element name="Freq500Hz" type="float"/>
              <element name="Freq1000Hz" type="float"/>
              <element name="Freq1500Hz" type="float" minOccurs="0"/>
              <element name="Freq2000Hz" type="float"/>
              <element name="Freq3000Hz" type="float" minOccurs="0"/>
              <element name="Freq4000Hz" type="float"/>
              <element name="Freq6000Hz" type="float" minOccurs="0"/>
              <element name="Freq8000Hz" type="float"/>
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################## -->
  <!--  Definition of VisualImpairment    -->
  <!-- ################################## -->
  <complexType name="VisualImpairmentType">
    <complexContent>
      <extension base="dia:UserCharacteristicBaseType">
          <sequence>
              <element name="Blindness" minOccurs="0">
                  <complexType>
                    <attribute name="eyeSide" use="required">
                        <simpleType>
                          <restriction base="NMTOKEN">
                            <enumeration value="both"/>
                            <enumeration value="left"/>
                            <enumeration value="right"/>
                          </restriction>
                        </simpleType>
                    </attribute>
                  </complexType>
              </element>
              <element name="LowVisionSymptoms" type="dia:LowVisionImpairmentType" minOccurs="0"/>
              <element name="ColorVisionDeficiency" type="dia:ColorVisionDeficiencyType" minOccurs="0"/>
          </sequence>
          <attribute name="rightSight" type="float" use="optional"/>
          <attribute name="leftSight" type="float" use="optional"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="LowVisionImpairmentType">
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
              <element name="LossOfFineDetail" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="LackOfContrast" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="LightSensitivity" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="NeedOfLight" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="CenterVisionLoss" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="PeripheralVisionLoss" type="dia:VisualImpairmentDegreeType" minOccurs="0"/>
              <element name="Hemianopia" minOccurs="0">
                  <complexType>
                    <attribute name="side" use="required">
                        <simpleType>
                          <restriction base="NMTOKEN">
                            <enumeration value="left"/>
                            <enumeration value="right"/>
                          </restriction>
                        </simpleType>
                    </attribute>
```

```
                </complexType>
            </element>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="VisualImpairmentDegreeType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <choice>
            <element name="NumericDegree" type="mpeg7:zeroToOneType"/>
            <element name="TextualDegree">
                <simpleType>
                  <restriction base="token">
                      <enumeration value="Severe"/>
                      <enumeration value="Medium"/>
                      <enumeration value="Mild"/>
                  </restriction>
                </simpleType>
            </element>
        </choice>
    </extension>
  </complexContent>
</complexType>
<!-- ########################################   -->
<!--  Definition of ColorVisionDeficiency          -->
<!-- ########################################   -->
<complexType name="ColorVisionDeficiencyType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="DeficiencyType">
                <simpleType>
                  <restriction base="token">
                      <enumeration value="Red-Deficiency"/>
                      <enumeration value="Green-Deficiency"/>
                      <enumeration value="Blue-Deficiency"/>
                      <enumeration value="CompleteColorBlindness"/>
                  </restriction>
                </simpleType>
            </element>
            <element name="DeficiencyDegree" type="dia:VisualImpairmentDegreeType"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of MobilityCharacteristics         -->
<!-- ################################################   -->
<complexType name="MobilityCharacteristicsType">
  <complexContent>
    <extension base="dia:UserCharacteristicBaseType">
        <sequence>
            <element name="UpdateInterval" type="dia:UpdateIntervalType" minOccurs="0"/>
            <element name="Directivity" type="dia:DirectivityType" minOccurs="0"/>
            <element name="Erraticity" type="dia:ErraticityType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="UpdateIntervalType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
            <element name="LastUpdatePoint" type="mpeg7:GeographicPointType" minOccurs="0"/>
            <element name="LastUpdateBinIndex" type="integer" minOccurs="0"/>
            <element name="LastUpdateTime" type="mpeg7:TimeType" minOccurs="0"/>
```

```
            <element name="Lmax" type="integer" minOccurs="0"/>
            <element name="Values" minOccurs="0">
              <simpleType>
                <restriction base="mpeg7:probabilityVector">
                  <length value="32"/>
                </restriction>
              </simpleType>
            </element>
          </sequence>
          <attribute name="xRadius" type="integer" use="optional"/>
          <attribute name="yRadius" type="integer" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="DirectivityType">
      <complexContent>
        <extension base="dia:DIABaseType">
          <sequence>
            <element name="Mean" type="float" minOccurs="0"/>
            <element name="Variance" type="float" minOccurs="0"/>
            <element name="Values" minOccurs="0">
              <simpleType>
                <restriction base="mpeg7:probabilityVector">
                  <length value="16"/>
                </restriction>
              </simpleType>
            </element>
          </sequence>
          <attribute name="measuredInterval" type="integer" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="ErraticityType">
      <complexContent>
        <extension base="dia:DIABaseType">
          <sequence>
            <element name="Values" minOccurs="0">
              <simpleType>
                <restriction base="mpeg7:probabilityVector">
                  <length value="128"/>
                </restriction>
              </simpleType>
            </element>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <!-- ########################################  -->
    <!--  Definition of Destination             -->
    <!-- ########################################  -->
    <complexType name="DestinationType">
      <complexContent>
        <extension base="dia:UserCharacteristicBaseType">
          <sequence>
            <element name="Time" type="mpeg7:TimeType" minOccurs="0"/>
            <element name="Location" type="mpeg7:PlaceType" minOccurs="0"/>
            <element name="DestinationClass" minOccurs="0">
              <complexType>
                <choice>
                  <element name="FreeClass" type="mpeg7:TextualType" minOccurs="0" maxOccurs="unbounded"/>
                  <element name="StereotypedClass" type="mpeg7:ControlledTermUseType"/>
                </choice>
              </complexType>
            </element>
            <element name="DestinationName" type="mpeg7:TextualType" minOccurs="0"/>
          </sequence>
        </extension>
```

```
    </complexContent>
  </complexType>
  <!-- ##################################################### -->
  <!--  Definition of Axmedis User      -->
  <!-- #####################################################-->
  <complexType name="AxmedisUser">
    <complexContent>
      <extension base="dia:UserType"/>
    </complexContent>
  </complexType>
```

11.13.2 Technical and Installation information

The JS class comes in the form of a static LIB. The init method has to be called in the source code where the Engine is initialized. This is necessary to add the set of functions to the Javascript Engine.

References to other major components needed	
Problems not solved	•
Configuration and execution context	

11.13.3 Draft User Manual

Exposed Methods

UserProfile()
Constructor for Initialising the Profile

loadXmlFile(String uri)
Loading the Xml from file

loadXmlString(String XmlString)
Loading Xml from an Xml String

createXmlfile(String uri)
Create and Xml file where the file is specified as uri

String getXml ()
Returns the contents of the XML file in String form

setXml (String XmlString)
Write the string as the contents of the XML file

String getAttribute(String DomPath, String Attribute)
It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix

setAttribute(String DomPath, String Attribute, String value)
Sets an individual values to the attributes of the xml file

getValue(String DomPath)
It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix

setValue (String DomPath, String value)

Sets an individual values to the elements of the xml file

createElement(String DomPath, String ElementName)
Create an xml element using the DomPath and Element Name

deleteElement(String DomPath)
Delete an xml element based on the DomPath

close()
Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.13.4 Examples of usage

Sample Java Script code to get the value of a user-profile element and set it with user specified value.

```
var userprofile= new UserProfile();  //  Initialises the User Profile
userprofile.loadXmlFile("C:\axmedis\testProfile.xml"); // Loads an xml file into the profile class

// Get the volume preference of the user
var volume= userprofile.getValue("Preferences/AudioPreferences /VolumeControl");

//set the volume preference of the user
userprofile.setValue("Preferences/AudioPreferences /VolumeControl","0.5");

//Close the file
userprofile.close();
```

11.13.5 Integration and compilation issues

None

11.14 JSDeviceProfile (UR)

Module Profile	
JSDeviceProfile	
Responsible Name	Badii
Responsible Partner	UR (previously IRC)
Status (proposed/approved)	Approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Library
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/profiling/ https://cvs.axmedis.org/repos/Framework/source /jsaxclasses/profiling/
Reference to the AXFW location of the demonstrator executable tool for internal	

download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	no	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
	Java Script	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK,

		proprietary, authorized or not
Xerces	Xerces 2.7.0	LGPL license
DELI	deli-0.9.7	BSD License
WURFL	WURFL-2.0.1	http://wurfl.sourceforge.net/licence.php
JS API Spidermonkey	JS API v.1.5	LGPL license

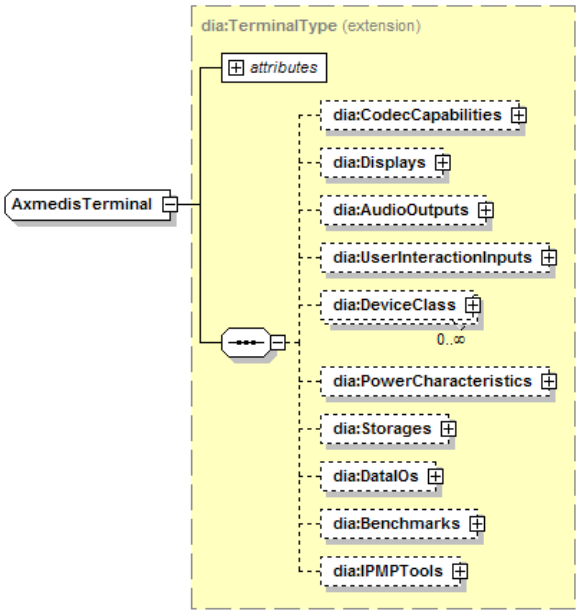
Device profile captures information related to user's device, that the service provider must take into account while delivering the contents. Generally the minimal set of elements required are those that affect the proper utilization of the contents on the device e.g. screen resolution for images. This profile represents the characteristic of the user's personal device and hence contains most vital information for the content providers that can be used to transcode the content to match the device characteristic and can be better utilized by the user. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-schema.xsd The JSDeviceProfile is a JS class to automatically create and manage Device profiles as a part of rule scripts for AXCP engine.

The functionalities and values for the device profiles can be accessed through the methods given below. These functions are only wrappers for the functions in the c++ class DeviceProfile.

- *DeviceProfile:* Constructor for Initialising the Profile
- *loadXmlFile:* Loading the Xml from file
- *loadXmlString:* Loading Xml from an Xml String
- *createXmlfile:*Create and Xml file where the file is specified as uri
- *getXml:* Returns the contents of the XML file in String form
- *setXml:* Write the string as the contents of the XML file
- *getAttribute:* It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setAttribute:* Sets an individual values to the attributes of the xml file
- *getValue:* It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setValue:* Sets an individual values to the elements of the xml file
- *createElement:* Create an xml element using the DomPath and Element Name
- *deleteElement:* Delete an xml element based on the DomPath
- *close*: Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.14.1 Module Design in terms of Classes

The device profile is a valid XML file based on following Schema. For entire schema on Device Profile please refer to the Axmedis-schema.xsd file.




```
<!-- ################################################   -->
<!--  Definition of Terminals          -->
<!-- ################################################   -->
<complexType name="TerminalsType">
  <complexContent>
    <extension base="dia:UsageEnvironmentPropertyBaseType">
        <sequence>
          <element name="Terminal" type="dia:TerminalType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of Terminal          -->
<!-- ################################################   -->
<complexType name="TerminalType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
          <element name="CodecCapabilities" type="dia:CodecCapabilitiesType" minOccurs="0"/>
          <element name="Displays" type="dia:DisplaysType" minOccurs="0"/>
          <element name="AudioOutputs" type="dia:AudioOutputsType" minOccurs="0"/>
          <element name="UserInteractionInputs" type="dia:UserInteractionInputsType" minOccurs="0"/>
          <element name="DeviceClass" type="dia:DeviceClassType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="PowerCharacteristics" type="dia:PowerCharacteristicsType" minOccurs="0"/>
          <element name="Storages" type="dia:StoragesType" minOccurs="0"/>
          <element name="DataIOs" type="dia:DataIOsType" minOccurs="0"/>
          <element name="Benchmarks" type="dia:BenchmarksType" minOccurs="0"/>
          <element name="IPMPTools" type="dia:IPMPToolsType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="TerminalCapabilityBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIABaseType"/>
  </complexContent>
</complexType>
```

```
<!-- ################################################   -->
<!--  Definition of CodecCapabilities    -->
<!-- ################################################   -->
<complexType name="CodecCapabilitiesType">
  <complexContent>
    <extension base="dia:TerminalCapabilityBaseType">
        <sequence>
          <element name="Decoding" type="dia:CodecCapabilityBaseType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="Encoding" type="dia:CodecCapabilityBaseType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of CodecCapability       -->
<!-- ################################################   -->
<complexType name="CodecCapabilityBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence minOccurs="0" maxOccurs="unbounded">
          <element name="Format" type="mpeg7:ControlledTermUseType"/>
          <element name="CodecParameter" type="dia:CodecParameterBaseType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AudioCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<complexType name="GraphicsCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<complexType name="ImageCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<complexType name="SceneGraphCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<complexType name="TransportCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<complexType name="VideoCapabilitiesType">
  <complexContent>
    <extension base="dia:CodecCapabilityBaseType"/>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of CodecParameter       -->
<!-- ################################################   -->
<complexType name="CodecParameterBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIABaseType"/>
  </complexContent>
</complexType>
<complexType name="CodecParameterBufferSizeType">
  <complexContent>
    <extension base="dia:CodecParameterBaseType">
```

```
            <sequence>
              <element name="BufferSize" type="integer" minOccurs="0"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="CodecParameterBitRateType">
      <complexContent>
        <extension base="dia:CodecParameterBaseType">
            <sequence>
              <element name="BitRate" minOccurs="0">
                  <complexType>
                      <simpleContent>
                          <extension base="nonNegativeInteger">
                              <attribute name="average" type="nonNegativeInteger" use="optional"/>
                              <attribute name="maximum" type="nonNegativeInteger" use="optional"/>
                          </extension>
                      </simpleContent>
                  </complexType>
              </element>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="CodecParameterMemoryBandwidthType">
      <complexContent>
        <extension base="dia:CodecParameterBaseType">
            <sequence>
              <element name="MemoryBandwidth" type="integer" minOccurs="0"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="CodecParameterVertexRateType">
      <complexContent>
        <extension base="dia:CodecParameterBaseType">
            <sequence>
              <element name="VertexRate" type="integer" minOccurs="0"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="CodecParameterFillRateType">
      <complexContent>
        <extension base="dia:CodecParameterBaseType">
            <sequence>
              <element name="FillRate" type="integer" minOccurs="0"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################   -->
    <!--  Definition of Displays      -->
    <!-- ################################################   -->
    <complexType name="DisplaysType">
      <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
              <element name="Display" type="dia:DisplayType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################   -->
    <!--  Definition of Display        -->
    <!-- ################################################   -->
    <complexType name="DisplayType">
```

```
    <complexContent>
      <extension base="dia:DIABaseType">
          <sequence>
            <element name="DisplayCapability" type="dia:DisplayCapabilityBaseType" minOccurs="0"
            maxOccurs="unbounded"/>
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DisplayCapabilityBaseType" abstract="true">
    <complexContent>
      <extension base="dia:DIABaseType"/>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of DisplayCapability     -->
  <!-- ################################################   -->
  <complexType name="DisplayCapabilityType">
    <complexContent>
      <extension base="dia:DisplayCapabilityBaseType">
          <sequence>
            <element name="Mode" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                    <sequence>
                        <element name="Resolution" type="dia:ResolutionType" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="SizeChar" minOccurs="0">
                            <complexType>
                              <attribute name="horizontal" type="integer" use="required"/>
                              <attribute name="vertical" type="integer" use="required"/>
                            </complexType>
                        </element>
                    </sequence>
                    <attribute name="refreshRate" type="float" use="optional"/>
                </complexType>
            </element>
            <element name="ScreenSize" minOccurs="0">
                <complexType>
                    <attribute name="horizontal" type="float" use="required"/>
                    <attribute name="vertical" type="float" use="required"/>
                </complexType>
            </element>
            <element name="ColorBitDepth" minOccurs="0">
                <complexType>
                    <attribute name="red" type="integer" use="required"/>
                    <attribute name="green" type="integer" use="required"/>
                    <attribute name="blue" type="integer" use="required"/>
                </complexType>
            </element>
            <element name="ColorPrimaries" minOccurs="0">
                <complexType>
                    <sequence>
                        <element name="ChromaticityRed" type="dia:ChromaticityType"/>
                        <element name="ChromaticityGreen" type="dia:ChromaticityType"/>
                        <element name="ChromaticityBlue" type="dia:ChromaticityType"/>
                        <element name="ChromaticityWhite" type="dia:ChromaticityType" minOccurs="0"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CharacterSetCode" type="mpeg7:characterSetCode" minOccurs="0" maxOccurs="unbounded"/>
            <element name="RenderingFormat" type="mpeg7:ControlledTermUseType" minOccurs="0"
            maxOccurs="unbounded"/>
          </sequence>
          <attribute name="stereoscopic" type="boolean" use="optional"/>
          <attribute name="maximumBrightness" type="float" use="optional"/>
          <attribute name="contrastRatio" type="positiveInteger" use="optional"/>
          <attribute name="gamma" type="float" use="optional"/>
          <attribute name="bitsPerPixel" type="integer" use="optional"/>
```

```
            <attribute name="colorCapable" type="boolean" use="optional"/>
            <attribute name="sRGB" type="boolean" use="optional"/>
            <attribute name="fieldSequentialColor" type="boolean" use="optional"/>
            <attribute name="backlightLuminance" type="mpeg7:zeroToOneType" use="optional"/>
            <attribute name="dotPitch" type="float" use="optional"/>
            <attribute name="activeDisplay" type="boolean" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<complexType name="ResolutionType">
    <complexContent>
        <extension base="dia:DIABaseType">
            <attribute name="horizontal" type="integer" use="required"/>
            <attribute name="vertical" type="integer" use="required"/>
            <attribute name="activeResolution" type="boolean" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<complexType name="ChromaticityType">
    <complexContent>
        <extension base="dia:DIABaseType">
            <attribute name="x" type="mpeg7:zeroToOneType" use="required"/>
            <attribute name="y" type="mpeg7:zeroToOneType" use="required"/>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of AudioOutputs       -->
<!-- ################################################   -->
<complexType name="AudioOutputsType">
    <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
                <element name="AudioOutput" type="dia:AudioOutputType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of AudioOutput         -->
<!-- ################################################   -->
<complexType name="AudioOutputType">
    <complexContent>
        <extension base="dia:DIABaseType">
            <sequence>
                <element name="AudioOutputCapability" type="dia:AudioOutputCapabilityBaseType" minOccurs="0"
                maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="AudioOutputCapabilityBaseType" abstract="true">
    <complexContent>
        <extension base="dia:DIABaseType"/>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of AudioOutputCapabilities       -->
<!-- ################################################   -->
<complexType name="AudioOutputCapabilitiesType">
    <complexContent>
        <extension base="dia:AudioOutputCapabilityBaseType">
            <sequence>
                <element name="Mode" type="dia:AudioModeType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="lowFrequency" type="float" use="optional"/>
            <attribute name="highFrequency" type="float" use="optional"/>
```

```
            <attribute name="signalNoiseRatio" type="float" use="optional"/>
            <attribute name="power" type="float" use="optional"/>
            <attribute name="numChannels" type="nonNegativeInteger" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="AudioModeType">
      <complexContent>
        <extension base="dia:DIABaseType">
            <attribute name="samplingFrequency" type="float" use="optional"/>
            <attribute name="bitsPerSample" type="integer" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################    -->
    <!--  UserInteractionInputs      -->
    <!-- ################################################    -->
    <complexType name="UserInteractionInputsType">
      <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
              <element name="UserInteractionInput" type="dia:UserInteractionInputType" minOccurs="0"
              maxOccurs="unbounded"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################   -->
    <!--  UserInteractionInputType     -->
    <!-- ################################################   -->
    <complexType name="UserInteractionInputType">
      <complexContent>
        <extension base="dia:DIABaseType">
            <sequence>
              <element name="UserInteractionInputSupport" type="dia:UserInteractionInputSupportBaseType" minOccurs="0"
              maxOccurs="unbounded"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="UserInteractionInputSupportBaseType" abstract="true">
      <complexContent>
        <extension base="dia:DIABaseType"/>
      </complexContent>
    </complexType>
    <complexType name="StringInputType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType"/>
      </complexContent>
    </complexType>
    <complexType name="KeyInputType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType">
            <sequence>
              <element name="KeyInput" type="mpeg7:ControlledTermUseType"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="MicrophoneType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType"/>
      </complexContent>
    </complexType>
    <complexType name="MouseType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType">
```

```
            <sequence>
              <element name="Mouse" type="dia:GenericMouseType"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="TrackballType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType">
            <sequence>
              <element name="Trackball" type="dia:GenericMouseType"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="PenType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType">
            <sequence>
              <element name="Pen" type="dia:GenericPenType"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="TabletType">
      <complexContent>
        <extension base="dia:UserInteractionInputSupportBaseType">
            <sequence>
              <element name="Tablet" type="dia:GenericPenType"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="GenericMouseType">
      <complexContent>
        <extension base="dia:DIABaseType">
            <attribute name="resolution" type="nonNegativeInteger" use="optional"/>
            <attribute name="buttons" type="nonNegativeInteger" use="required"/>
            <attribute name="scrollWheel" type="boolean" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="GenericPenType">
      <complexContent>
        <extension base="dia:DIABaseType">
            <attribute name="resolution" type="nonNegativeInteger" use="optional"/>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################   -->
    <!--  Definition of DeviceClass     -->
    <!-- ################################################   -->
    <complexType name="DeviceClassType">
      <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
              <element name="DeviceClass" type="mpeg7:ControlledTermUseType" minOccurs="0"/>
            </sequence>
        </extension>
      </complexContent>
    </complexType>
    <!-- ################################################   -->
    <!--  Definition of PowerCharacteristics            -->
    <!-- ################################################   -->
    <complexType name="PowerCharacteristicsType">
      <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
```

```
            <attribute name="averageAmpereConsumption" type="integer" use="optional"/>
            <attribute name="batteryCapacityRemaining" type="integer" use="optional"/>
            <attribute name="batteryTimeRemaining" type="integer" use="optional"/>
            <attribute name="runningOnBatteries" type="boolean" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of Storages      -->
<!-- ################################################   -->
<complexType name="StoragesType">
    <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
                <element name="Storage" type="dia:StorageType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of Storage        -->
<!-- ################################################   -->
<complexType name="StorageType">
    <complexContent>
        <extension base="dia:DIABaseType">
            <sequence>
                <element name="StorageCharacteristic" type="dia:StorageCharacteristicBaseType" minOccurs="0"
                maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="StorageCharacteristicBaseType" abstract="true">
    <complexContent>
        <extension base="dia:DIABaseType"/>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of StorageCharacteristics          -->
<!-- ################################################   -->
<complexType name="StorageCharacteristicsType">
    <complexContent>
        <extension base="dia:StorageCharacteristicBaseType">
            <attribute name="inputTransferRate" type="nonNegativeInteger" use="optional"/>
            <attribute name="outputTransferRate" type="nonNegativeInteger" use="optional"/>
            <attribute name="size" type="float" use="optional"/>
            <attribute name="writable" type="boolean" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of DataIOs      -->
<!-- ################################################   -->
<complexType name="DataIOsType">
    <complexContent>
        <extension base="dia:TerminalCapabilityBaseType">
            <sequence>
                <element name="DataIO" type="dia:DataIOType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of DataIO        -->
<!-- ################################################   -->
<complexType name="DataIOType">
    <complexContent>
```

```
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="DataIOCharacteristic" type="dia:DataIOCharacteristicBaseType" minOccurs="0"
          maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DataIOCharacteristicBaseType" abstract="true">
    <complexContent>
      <extension base="dia:DIABaseType"/>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of DataIOCharacteristics           -->
  <!-- ################################################   -->
  <complexType name="DataIOCharacteristicsType">
    <complexContent>
      <extension base="dia:DataIOCharacteristicBaseType">
        <attribute name="busWidth" type="nonNegativeInteger" use="optional"/>
        <attribute name="transferSpeed" type="nonNegativeInteger" use="optional"/>
        <attribute name="maxDevices" type="nonNegativeInteger" use="optional"/>
        <attribute name="numDevices" type="nonNegativeInteger" use="optional"/>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of Benchmarks                      -->
  <!-- ################################################   -->
  <complexType name="BenchmarksType">
    <complexContent>
      <extension base="dia:TerminalCapabilityBaseType">
        <sequence>
          <element name="Benchmark" type="dia:BenchmarkType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of Benchmark                       -->
  <!-- ################################################   -->
  <complexType name="BenchmarkType">
    <complexContent>
      <extension base="dia:DIABaseType">
        <sequence>
          <element name="DeviceBenchmark" type="dia:DeviceBenchmarkBaseType" minOccurs="0"
          maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  Definition of DeviceBenchmarkBaseType                    -->
  <!-- ################################################ -->
  <complexType name="DeviceBenchmarkBaseType" abstract="true">
    <complexContent>
      <extension base="dia:DIABaseType"/>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  Definition of CPUBenchmarkType                -->
  <!-- ################################################ -->
  <complexType name="CPUBenchmarkType">
    <complexContent>
      <extension base="dia:DeviceBenchmarkBaseType">
        <attribute name="name" type="mpeg7:termReferenceType" use="required"/>
        <attribute name="baseValue" type="float" use="required"/>
```

```
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  Definition of ThreeDBenchmarkType               -->
  <!-- ################################################ -->
  <complexType name="ThreeDBenchmarkType">
    <complexContent>
      <extension base="dia:DeviceBenchmarkBaseType">
          <attribute name="name" type="mpeg7:termReferenceType" use="required"/>
          <attribute name="meanValue" type="float" use="required"/>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################   -->
  <!--  Definition of IPMPTools       -->
  <!-- ################################################   -->
  <complexType name="IPMPToolsType">
    <complexContent>
      <extension base="dia:TerminalCapabilityBaseType">
          <sequence>
            <element name="IPMPTool" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                    <simpleContent>
                        <extension base="base64Binary">
                            <attribute name="ToolCategory" use="required">
                              <simpleType>
                                  <restriction base="string">
                                    <enumeration value="MPEG2"/>
                                    <enumeration value="MPEG4"/>
                                  </restriction>
                              </simpleType>
                            </attribute>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="IPMPSType" type="hexBinary" minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################           -->
  <!--  Definition of Axmedis Terminal              -->
  <!-- ################################################-->
  <complexType name="AxmedisTerminal">
    <complexContent>
      <extension base="dia:TerminalType"/>
    </complexContent>
  </complexType>
```

11.14.2 Technical and Installation information

The JS class comes in the form of a static LIB. The init method has to be called in the source code where the Engine is initialized. This is necessary to add the set of functions to the Javascript Engine.

References to other major components needed	
Problems not solved	•
Configuration and execution context	

11.14.3 Draft User Manual

DeviceProfile()
Constructor for Initialising the Profile

loadXmlFile(String uri)
Loading the Xml from file

loadXmlString(String XmlString)
Loading Xml from an Xml String

createXmlfile(String uri)
Create and Xml file where the file is specified as uri

getXml ()
Returns the contents of the XML file in String form

setXml (String XmlString)
Write the string as the contents of the XML file

getAttribute(String DomPath, String Attribute)
It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix

setAttribute(String DomPath, String Attribute, String value)
Sets an individual values to the attributes of the xml file

getValue(String DomPath)
It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix

setValue (String DomPath, String value)
Sets an individual values to the elements of the xml file

createElement(String DomPath, String ElementName)
Create an xml element using the DomPath and Element Name

deleteElement(String DomPath)
Delete an xml element based on the DomPath

close()
Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.14.4 Examples of usage

Sample code to demonstrate copying one device profile into another file is given below. This example is mainly to demonstrate the methods used for retrieving the xml contents from a file and writing it to another file.

```
var deviceprofile1= new DeviceProfile();  //  Initialises the Device Profile for the source xml file
var deviceprofile2= new DeviceProfile();  //  Initialises the Device Profile for the destination  xml file

// Loads an source xml file
deviceprofile1.loadXmlFile("C:\axmedis\sourceProfile.xml");
```

```
// Retrieves the xml-string from the xml file and stores it into the variable xml
var xml=deviceprofile1.getXml();


//Creates the destination xml file
deviceprofile2.createXmlfile("C:\axmedis\destProfile.xml")
// Sets the contents for the destination xml file
deviceprofile2.setXml(xml)

// Closing both the xml files
deviceprofile1.close();
deviceprofile2.close();
```

11.14.5 Integration and compilation issues

none

11.15 JSNetworkProfile (UR)

Module Profile	
JSNetworkProfile	
Responsible Name	Badii
Responsible Partner	UR (previously IRC)
Status (proposed/approved)	approved
Implemented/not implemented	Implemented
Status of the implementation	100%
Executable or Library/module (Support)	Library
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows, Linux, Mac OS X
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/profiling/ https://cvs.axmedis.org/repos/Framework/source /jsaxclasses/profiling/
Reference to the AXFW location of the demonstrator executable tool for internal download	
Reference to the AXFW location of the demonstrator executable tool for public download	
Address for accessing to WebServices if any, add accession information (user and Passwd) if any	
Test cases (present/absent)	
Test cases location	http://////////////////
Usage of the AXMEDIS configuration manager (yes/no)	No
Usage of the AXMEDIS Error Manager (yes/no)	No
Major Problems not solved	--

	--	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
	Java Script	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces	Xerces 2.7.0	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL license

Network profile captures information related to the carrier that is used for delivery of the content to the user's device. It is necessary for the distributors to know various characteristics of the Network so as to provide the promised QoS. The information contained in this profile can also be used to for transcoding of the contents to better utilize both network and device capabilities. The elements in this profile are the minimal set of elements that is required to meet the AXMEDIS User's personalization needs and to meet the distributor's needs in terms of the network capabilities. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-schema.xsd.

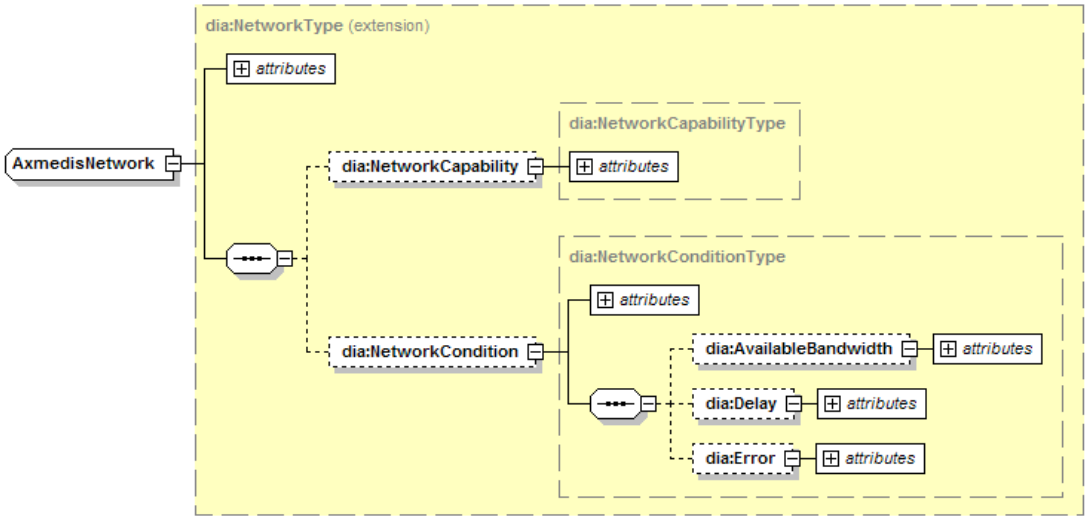
The JSNetwworkProfile is a JS class to automatically create and manage user profiles as a part of rule scripts for AXCP engine.

JS_NetworkProfile updates/retrieves the Network Profile stored in the database. It uses the following functions:

- *NetworkProfile:* Constructor for Initialising the Profile
- *loadXmlFile:* Loading the Xml from file
- *loadXmlString:* Loading Xml from an Xml String
- *createXmlfile:*Create and Xml file where the file is specified as uri
- *getXml:* Returns the contents of the XML file in String form
- *setXml:* Write the string as the contents of the XML file
- *getAttribute:* It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setAttribute:* Sets an individual values to the attributes of the xml file
- *getValue:* It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix
- *setValue:* Sets an individual values to the elements of the xml file
- *createElement:* Create an xml element using the DomPath and Element Name
- *deleteElement:* Delete an xml element based on the DomPath
- *close*: Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.15.1 Module Design in terms of Classes

The Network profile is a valid XML file based on following Schema. Please refer to axmedis-network-profile.xsd for entire schema file.




```
<!-- ################################################   -->
<!--  Definition of Networks                    -->
<!-- ################################################   -->
<complexType name="NetworksType">
  <complexContent>
    <extension base="dia:UsageEnvironmentPropertyBaseType">
        <sequence>
          <element name="Network" type="dia:NetworkType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################   -->
<!--  Definition of Network      -->
<!-- ################################################   -->
<complexType name="NetworkType">
  <complexContent>
    <extension base="dia:DIABaseType">
        <sequence>
          <element name="NetworkCapability" type="dia:NetworkCapabilityType" minOccurs="0"/>
          <element name="NetworkCondition" type="dia:NetworkConditionType" minOccurs="0"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="NetworkCharacteristicBaseType" abstract="true">
  <complexContent>
    <extension base="dia:DIABaseType"/>
  </complexContent>
</complexType>
<!-- ########################################   -->
<!--  Definition of NetworkCapability               -->
<!-- ########################################   -->
<complexType name="NetworkCapabilityType">
  <complexContent>
    <extension base="dia:NetworkCharacteristicBaseType">
        <attribute name="maxCapacity" type="nonNegativeInteger" use="optional"/>
        <attribute name="minGuaranteed" type="nonNegativeInteger" use="optional"/>
        <attribute name="inSequenceDelivery" type="boolean" use="optional"/>
        <attribute name="errorDelivery" type="boolean" use="optional"/>
        <attribute name="errorCorrection" type="boolean" use="optional"/>
    </extension>
  </complexContent>
</complexType>
<!-- ########################################   -->
<!--  Definition of NetworkCondition                -->
<!-- ########################################   -->
```

```
<complexType name="NetworkConditionType">
  <complexContent>
    <extension base="dia:NetworkCharacteristicBaseType">
        <sequence>
          <element name="AvailableBandwidth" minOccurs="0">
            <complexType>
                <attribute name="minimum" type="nonNegativeInteger" use="optional"/>
                <attribute name="maximum" type="nonNegativeInteger" use="optional"/>
                <attribute name="average" type="nonNegativeInteger" use="optional"/>
            </complexType>
          </element>
          <element name="Delay" minOccurs="0">
            <complexType>
                <attribute name="packetTwoWay" type="nonNegativeInteger" use="optional"/>
                <attribute name="packetOneWay" type="nonNegativeInteger" use="optional"/>
                <attribute name="delayVariation" type="integer" use="optional"/>
            </complexType>
          </element>
          <element name="Error" minOccurs="0">
            <complexType>
                <attribute name="packetLossRate" type="mpeg7:nonNegativeReal" use="optional"/>
                <attribute name="bitErrorRate" type="nonNegativeInteger" use="optional"/>
            </complexType>
          </element>
        </sequence>
        <attribute name="startTime" type="mpeg7:timePointType" use="optional"/>
        <attribute name="duration" type="mpeg7:durationType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
<!-- ################################################            -->
<!--  Definition of Axmedis Network -->
<!-- ################################################-->
<complexType name="AxmedisNetwork">
  <complexContent>
    <extension base="dia:NetworkType"/>
  </complexContent>
</complexType>
```

11.15.2 Technical and Installation information

The JS class comes in the form of a static LIB. The init method has to be called in the source code where the Engine is initialized. This is necessary to add the set of functions to the Javascript Engine.

References to other major components needed	
Problems not solved	•
Configuration and execution context	

11.15.3 Draft User Manual

NetworkProfile()

Constructor for Initialising the Profile

loadXmlFile(String uri)

Loading the Xml from file

loadXmlString(String XmlString)

Loading Xml from an Xml String

createXmlfile(String uri)

Create and Xml file where the file is specified as uri

getXml ()
Returns the contents of the XML file in String form

setXml (String XmlString)
Write the string as the contents of the XML file

getAttribute(String DomPath, String Attribute)
It returns the corresponding value of the attributes from the xml file. Attributes for the respective profiles are specified in the Profiles documentation provided in the Appendix

setAttribute(String DomPath, String Attribute, String value)
Sets an individual values to the attributes of the xml file

getValue(String DomPath)
It returns the corresponding value of the element from the xml file. Possible Values for the respective profiles are specified in the Profiles documentation provided in the Appendix

setValue (String DomPath, String value)
Sets an individual values to the elements of the xml file

createElement(String DomPath, String ElementName)
Create an xml element using the DomPath and Element Name

deleteElement(String DomPath)
Delete an xml element based on the DomPath
close()
Closes the xml file. If any element is change then the changes are saved and then the xml file is closed.

11.15.4 Examples of usage

Sample Java Script code to get the value of a Network -profile attribute and set it with user specified value.

```
var networkprofile= new Network Profile();  //  Initialises the Network Profile
networkprofile.loadXmlFile("C:\axmedis\testProfile.xml"); // Loads an xml file into the profile class

// Get the yRadius
var yrad= networkprofile.getAttribute ("Mobility/UpdateInterval", "yRadius");

//set the yRadius
networkprofile.setAttribute("Mobility/UpdateInterval", "yRadius",0.5");

//Close the file
networkprofile.close();
```

11.15.5 Integration and compilation issues

None

11.16 JSAxDecisionTakingEngine (UR)

Module Profile		
JSAxDecisionTakingEngine		
Responsible Name	Badii	
Responsible Partner	UR (previously IRC)	
Status (proposed/approved)	approved	
Implemented/not implemented	Implemented	
Status of the implementation	100%	
Executable or Library/module (Support)	Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows, Linux, Mac OS X	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/ https://cvs.axmedis.org/repos/Framework/source /jsaxclasses/	
Reference to the AXFW location of the demonstrator executable tool for internal download		
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	No	
Usage of the AXMEDIS Error Manager (yes/no)	No	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section

Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
	Java Script	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Xerces	Xerces 2.7.0	LGPL license
JS API Spidermonkey	JS API v.1.5	LGPL license

Decision Taking Engine acquires information from the AXMEDIS profiles related to the user, the network and the device in use.. It processes this information to make important decisions to arrive at an adaptation strategy or criterion for the AXCP adaptation rules to follow and derive suitably adapted objects from simple and composite objects. The JSAxDecisionTakingEngine is a JS class to automatically instantiate and activate the decision engine to process the profiles as a part of the adaptation rule scripts for AXCP engine. These scripts are specified in detail under section 11.16.4.2.

JSAxDecisionTakingEngine processes the profiles to derive the adaptation strategy for DIA processing (object daptation). It uses the following functions:

- *AxDecisionTakingEngine:* Constructor for Initialising the engine

- *Activate:* Activate decision taking engine using the profiles available at the specified paths

11.16.1 Module Design Details

The JSAxDecisionTakingEngine module is an interface to the AXCP with the two exposed functions listed in the previous section. Internally, this module uses complex functionalities packaged as a Constraint Engine for Digital Item Adaptation using Constraint Satisfaction based on Gecode.

Constraint satisfaction

Constraint satisfaction is a well-used technique in artificial intelligence to find solutions to a set of constraints involving variables whose domains are usually finite. Formally, a constraint satisfaction problem (CSP) is a triple $\langle X,C,D\rangle$, where X ranges over variables in the problem, D is the domain of these variables (usually finite in practise) and C is a set of constraints that must be satisfied. The solution to a CSP, if any, is a set of possible values of variables in X, whose values are chosen from D and which satisfy C. Specifically,

for problems over finite domains the solutions are found by searching using many advanced techniques to find some or all the solutions to a CSP efficiently.

Within the AXMEDIS context, the variables *X* correspond to parameters of interest in the AXMEDIS resource description, and the relevant AXMEDIS profiles which must guide the resource adaptation. For example, suppose that a video resource to be rendered on a device must have a given aspect ratio in order to maximise the use of screen space, and must not exceed certain storage space then aspect ratio and storage requirements would be variables in the CSP. Additionally, the relationship between storage space and storage space could be part of the constraints *C* the the problem being modelled.

Gecode

There are families of languages, called constraint programming languages which provide programmatic frameworks for solving CSPs. Since AXMEDIS uses mainly the C++ programming language, Gecode (*http://www.gecode.org/*) is an open source framework which provides C++ libraries for solving CSPs can be adopted.

Gecode licence exists as MIT license and applies as follows as per reported on the reference website (http://www.gecode.org/license.html):

"This software and its documentation are copyrighted by the individual authors as listed in each file. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE." (http://www.gecode.org/license.html)

11.16.1.1 Constraint Modelling

We consider the problem of digital resource adaptation in the AXMEDIS framework and propose a technique to its automation. In a typical scenario, we are presented with a digital resource which must be adapted to a target AXMEDIS object that must conform to a set of AXMEDIS profile specifications. These specifications can be considered as constraints guiding the adaptation process. Hence, the systematic selection of a solution space of feasible adaptations is amenable to the standard artificial intelligence technique of constraint satisfaction [**Errore. L'origine riferimento non è stata trovata.**, **Errore. L'origine riferimento non è stata trovata.**]. That is, the problem can be cast to a general constraint satisfaction problem, which provides us with generic tools to automate the discovery of solution spaces.

As a concrete example, an AXMEDIS video resource that must be rendered on a wireless mobile device with low resolution, small screen size and small computational capability already presents us with network constraints (wireless) and device-related constraints such as low computational power and low display quality. These information about the device and the network are available in the AXMEDIS device profile for that particular mobile device and networking profile associated with this device and video-streaming service. Additionally, the user of the mobile device may have preferences for which licenses he or she wishes to be associated with the video to be streamed to the device, as well as quality of service requirements. On the other hand, the video to be adapted itself may have licenses which restrict where and how it may be viewed. All these present constraints that must be satisfied in order to provide this video service in a manner that is acceptable to all the parties involved.

The key step in transforming the AXMEDIS resource adaptation problem to a form which can be fed to and automated by a CSP framework is the development of a model of the AXMEDIS world as a constraint satisfaction problem. Luckily, the AXMEDIS object models and profiles already are specified as XML schema which circumscribe the variables *X* that are used within the AXMEDIS framework. The domains *D* of these variables may also be extracted from these schemas. So the first stage of the analysis will involve the selection from the XML schema specifications, the relevant variables that will be needed for our analyses as well as defining their domain of definition. Some of these variables may not be numerical and thus may need auxiliary functions defined over them for the relevant operations that might be needed in the constraint satisfaction problem. For example, in the process of searching for a compatible licenses to a given specification, auxiliary functions must be defined that provides the necessary "intelligence" which determines when licenses are "compatible" and so on. On the other hand, partial ordering of preferences or the assignment of integer values to non-numeric attributes may need to be done in the modelling stage. Unless the AXMEDIS model changes, the CSP modelling stage need only be carried out once in the system development life-cycle (SDLC).

Once the modelling has been completed, rules and constraints specifying relationship among the identified variables can be specified. These can continue to be added to the system as more requirements arise and better heuristics are discovered. These rules may also be removed to relax constraints.

Input-Output and Software Specifications

Since the AXMEDIS framework uses the C++ programming language, the software system specification only consider relevant C++-based libraries for the system development. As mentioned earlier, Gecode is recommended as a CSP programming language. In addition, since XML-based documents and schemas must be processed, the Apache family of libraries for processing XML can be used. These may be found at *http://xerces.apache.org/xerces-c/*. These provide a set of validating XML parsers that are written in a portable subset of C++.

In the modelling stage of the SDLC, the XML APIs from the Apache foundation can be used to extracted the relevant CSP variables and domains from XML schemas. During the actual runtime itself, these libraries will also be useful in extracting data from XML-based AXMEDIS resource description and AXMEDIS profiles that must be passed to the CSP runtime.

The overall input-output specification of the framework is as follows:

- Modelling stage
    1. Input: - XML Schemas describing AXMEDIS Resources and Profiles
    2. Output: - Variables and Domains for the CSP (these may need to be manually edited)
    3. Auxiliary functions must be defined for operations non-numeric variables
    4. General Constraints and Rules written for the CSP problem.
- Runtime At this stage we find solutions to the CSP problem based on specific resource descriptions and profiles that are presented in XML form
    1. Input: - XML documents describing AXMEDIS Resources and Profiles
    2. Output: - Solution to the CSP problem describing sets of target/adapted resource that are valid for the given AXMEDIS resource and profiles. These were referred to as the adaptation-actionable sets in earlier documents.

11.16.1.2 Illustration

To illustrate the approach, suppose that the user of a mobile device wishes to display an uncropped image on the screen with size $\ell \times w$ (measured in inches). This device can display images at different resolutions **Errore.** in pixels per inch (assuming that the horizontal and vertical resolutions are the same). Furthermore, let us assume that this device supports several image formats such as JPEG, GIF, BMP, ...; and has wireless data connectivity modes over WIFI, Satellite, GPRS, ... These information are gathered from the AXMEDIS device profile. Depending on the network used, the cost to this user varies because the various network service providers charge based on the available badwidth and duration of transfer. Information about the supported bandwidth and the associated cost are available in the AXMEDIS profiles of the various supported networks.

Selecting adaptation strategies

We could ask many questions in the scenario, corresponding to various adaptation strategies which most meet the user's need. For example, the user may want to minimise the associated cost of consuming this service. In other words, we want to choose a network and image format combination which minimises cost. Or, the user may also desire to minimise the waiting time, while being able to view the highest possible resolution on the device at minimal cost. This may involve selecting an image format with efficient compression capability to minimise the size of the image to be transferred across the network. All these amount to a network of constraints which we want to solve in order to choose the appropriate adaptation strategy.

Modelling the system

We start off by illustrating the model development aspect of the SDLC. Once the model has been developed, we can then implement the system in any generic CSP environment. Further restrictions and refinement can be done via new constraints added to the system, or we might relax requirements by removing constraints.

The necessary parameters of our constraint satisfaction problem include the target screen dimensions ($\ell \times w$), the resolution **Errore.** to use, the image format $f \in \{JPEG, BMP, GIF, ...\}$, the network type $n \in \{internet, satellite, kiosk, ...\}$, for example. Let us assume that the file size (s) of the adapted image needs to be calculated. For simplicity, we assume that there is a constant, k_f, associated with each image format f describing the compression capability of that format; that is, a factor which translates the number of pixels to be encoded to a file size in that format (This is obviously a simplistic view of image compression technology employed by different formatting standards and is dependent on the image being encoded. In practise, the file size may just simply be measured directly from the encoded image.). Hence, we compute the file size of the image (under the format f) as follows:

$$s_f = \ell \times w \times r^2 \times k_f$$

Since the network service providers charge by download size and the duration of download, we need to compute in addition to the size of the image to be sent over the network, the duration (d_n) that it takes to send the file over the network n. This, again for simplicity, we assume is a linear function of the file size calculated as

Errore.

where d'_n is the network-related data transfer overheads for sending file of size s over network n and b_n is the associated data-rate (that is, "bandwidth") of the network. The parameters d'_n and b_n are derived or extracted from n's profile. Finally, let us assume that the cost c_n of transferring a file of size s within a duration of d_n over n is computed as

Errore.

where c'_n is some pricing factor for the network n.

In this example, we are just interested in checking what combination of parameters are possible, that is, the solution to this system without any additional constraints. As highlighted earlier, the goal might be to minimise or maximise the parameters identified above for cost efficiency or quality of service purposes. This could then be added incrementally to the system as additional constraints.

11.16.1.3 The AXMEDIS Constraint Solving Engine

We have developed a powerful constraint solving system for decision making in the AXMEDIS adaptation process. This engine uses Gecode internally for solving constraints. The key aspect of this is a modelling language, which we have developed for defining domains and expressing constraints in the AXMEDIS system. Details of this language are given in section **Errore. L'origine riferimento non è stata trovata.**. The key idea is that parameters in our adaptation strategy such as user preferences, device profiles and possible target adaptations of an AXMEDIS resource can be used in the development of a model which are passed to the engine in a fairly high-level language. This model is automatically solved and the solutions, if any, forms the basis of our decision for a particular resource adaptation strategy.

The engine does semi-symbolic computation, based on a model written in a very flexible and expressive AXMEDIS modelling language to find the set of solutions to an AXMEDIS constraint problem. The main entry point to this engine is a class called ***AXMEDISCSP***. This class has a static method which accepts a model and returns a vector of strings containing the solution(s) to the model. The interface is as follows:

static vector<string> AXMEDISCSP::solve(const char* model);

Clearly, this interface could be extended to accept data from arbitrary input streams such as files, URLs and so on, but we did not implement this. Another interesting extension would be to build the model directly from XML profiles and resource descriptions.

The AXMEDIS constraint solving engine performs subsymbolic analysis by using a model specification written in a fairly high-level modelling language that we developed. Since Gecode only deals with integer arithmetic, a lot of analysis needs to be performed upfront to translate the high-level specifications to a form which can be solved by Gecode. The model is parsed and we do a semantic analysis to check whether it conforms to the grammar of the AXMEDIS modelling language (please refer to Section **Errore. L'origine riferimento non è stata trovata.** for details). We then build up an internal object representation that can be processed by Gecode which is consequently solved. Note however that Gecode places some limitations on the kind of models that can be developed. For example, Gecode cannot be used to solve arbitrary mathematical expressions. However, we do not intend to extend the Gecode API for this reason, but we have simply used whatever is available to us. It must also be said that solving arbitrary equations and constraints is undecidable or may be simply intractable. As the language is quite expressive, hopefully the modelling language should be sufficient to meet the needs of AXMEDIS decision framework. However, feedback would be appreciated on specific constraints that cannot be expressed in the language.

We use GNU Flex and Bison to make a Lexical and Grammar analyser for AXMEDIS models. The result is a list of executable sentences which are then used to generate a Gecode model and solved. We discuss the grammar next and demonstrate its power by way of a simple example.

11.16.1.4 The AXMEDIS Constraint Modelling Language

We present the syntax of the AXMEDIS Constraint Modelling Language below using the BNF format. A model is a sequence of semicolon separated statements. The precise definition is as follows.

STATEMENT	:=	TYPEDEF | DEFINITION | CONSTRAINT
NUM	:=	([0-9])+
RANGE	:=	NUM : NUM
ID	:=	([a-zA-Z_])([a-zA-Z0-9_])*
DOMAIN	:=	[(NUM | RANGE)(, (NUM | RANGE))*]
op	:=	+ | - | *
rel	:=	= | ! = | < | <= | > | >=
EXP	:=	NUM | ID | EXP op EXP | (EXP)
INEQ	:=	EXP rel EXP
TYPEDEF	:=	typedef ID DOMAIN
DEFINITION	:=	ID DOMAIN | ID (,ID)* : ID
CONSTRAINT	:=	EXP INEQ EXP

The AXMEDIS Modelling Language

Defining variables, domains and expressing

Variables in a constraint modelling problem have associated domains of definition, which are usually finite as we have highlighted previously. We now discuss how to define variables and their associated domains under the AXMEDIS modelling framework.

Variables and identifiers (ID) in the modelling language are alphanumeric characters which start with either an alphabet or underscore and can contain any subsequent number of underscores and alphanumeric

characters. This flexibility allows us to use intuitive names for parameters in a model. Identifiers in the model are case sensitive. A domain of (integer) values is a comma-separated list of numbers or ranges, where a range is a colon separated pair of numbers and the range n:m denotes the set of integer values between n and m inclusive. Assume that t we want to define a variable x whose domain is say the set of values 1,2,3,87,7,8,9,10. Then this can be simply defined as x[1:3,87,7:10]. If there are several variables with the same domain, the typedef construct provides us with a construct to define that domain as a type, which can subsequently used in defining those variables. Let us say that there are other variables y and z which share the same domain as x above, we might instead define a domain X via the typedef construct as typedefX[1:3,87,7:10], and then followed by the statement x,y,z:X. This says that X is simply a domain (and is thus not treated as a variable), and that x,y and x have the domain of values of X.

Simple arithmetic expressions are supported as shown and these can be used to construct inequalities, specifying constraints on the allowed values of the variables in the model. These are parsed and models are automatically generated and solved with the results passed back to the user. While it is possible to limit the number of results returned in models which have multiple solutions, and also customise the strategy for searching for the solution; at the moment we we have simply explore the whole space and return all the solutions in the space.

Modelling the example

There is a sample executable file that uses the Modelling APIs developed to solve models submitted to it on the command line. The script shown in Figure **Errore. L'origine riferimento non è stata trovata.** below is a model of the example presented earlier where various constants have been assigned values and fictitious domains have been defined and assigned to the various model parameters. This illustrates the flexibility of the framework and the declarative nature of the solution.

```
typedef DIM[1:10]; l, w : DIM;
bandwidth[1:10, 20:30]; size [1:50]; resolution[1:10];
data_rate[3:5, 20]; cost[100:500];
size = l * w * resolution * resolution * 1;
data_rate * bandwidth = 3 * size;
cost = 3*size*data_rate;
```

A sample model

The solution from this program is shown below.

```
solution to the model::

typedef DIM[1:10]; l, w : DIM; bandwidth[1:10, 20:30];
 size [1:50]; resolution[1:10]; size = l * w * resolution * resolution * 1;
cost[100:500]; data_rate[3:5, 20]; cost = 3*size*data_rate;
data_rate * bandwidth = 3 * size;

_____________________________________________

No. solution ({l, w, bandwidth, size, resolution, cost, data_rate})
1.    {1, 1, 25, 25, 5, 225, 3}
2.    {1, 3, 27, 27, 3, 243, 3}
3.    {1, 5, 20, 20, 2, 180, 3}
4.    {1, 6, 24, 24, 2, 216, 3}
5.    {1, 7, 28, 28, 2, 252, 3}
6.    {2, 10, 20, 20, 1, 180, 3}
7.    {2, 3, 24, 24, 2, 216, 3}
8.    {3, 1, 27, 27, 3, 243, 3}
9.    {3, 7, 21, 21, 1, 189, 3}
10.   {3, 8, 24, 24, 1, 216, 3}
11.   {3, 9, 27, 27, 1, 243, 3}
```

12. {3, 10, 30, 30, 1, 270, 3}
13. {3, 2, 24, 24, 2, 216, 3}
14. {4, 5, 20, 20, 1, 180, 3}
15. {4, 6, 24, 24, 1, 216, 3}
16. {4, 7, 28, 28, 1, 252, 3}
17. {5, 1, 20, 20, 2, 180, 3}
18. {5, 4, 20, 20, 1, 180, 3}
19. {5, 5, 25, 25, 1, 225, 3}
20. {5, 6, 30, 30, 1, 270, 3}
21. {6, 1, 24, 24, 2, 216, 3}
22. {6, 4, 24, 24, 1, 216, 3}
23. {6, 5, 30, 30, 1, 270, 3}
24. {7, 1, 28, 28, 2, 252, 3}
25. {7, 3, 21, 21, 1, 189, 3}
26. {7, 4, 28, 28, 1, 252, 3}
27. {8, 3, 24, 24, 1, 216, 3}
28. {9, 3, 27, 27, 1, 243, 3}
29. {10, 2, 20, 20, 1, 180, 3}
30. {10, 3, 30, 30, 1, 270, 3}
31. {1, 1, 27, 36, 6, 432, 4}
32. {1, 2, 24, 32, 4, 384, 4}
33. {1, 3, 9, 12, 2, 144, 4}
34. {1, 4, 27, 36, 3, 432, 4}
35. {1, 7, 21, 28, 2, 336, 4}
36. {1, 8, 24, 32, 2, 384, 4}
37. {1, 9, 27, 36, 2, 432, 4}
38. {1, 10, 30, 40, 2, 480, 4}
39. {2, 1, 24, 32, 4, 384, 4}
40. {2, 2, 27, 36, 3, 432, 4}
41. {2, 6, 9, 12, 1, 144, 4}
42. {2, 4, 24, 32, 2, 384, 4}
43. {2, 5, 30, 40, 2, 480, 4}
44. {3, 1, 9, 12, 2, 144, 4}
45. {3, 4, 9, 12, 1, 144, 4}
46. {3, 3, 27, 36, 2, 432, 4}
47. {4, 1, 27, 36, 3, 432, 4}
48. {4, 3, 9, 12, 1, 144, 4}
49. {4, 7, 21, 28, 1, 336, 4}
50. {4, 8, 24, 32, 1, 384, 4}
51. {4, 9, 27, 36, 1, 432, 4}
52. {4, 10, 30, 40, 1, 480, 4}
53. {4, 2, 24, 32, 2, 384, 4}
54. {5, 8, 30, 40, 1, 480, 4}
55. {5, 2, 30, 40, 2, 480, 4}
56. {6, 2, 9, 12, 1, 144, 4}
57. {6, 6, 27, 36, 1, 432, 4}
58. {7, 1, 21, 28, 2, 336, 4}
59. {7, 4, 21, 28, 1, 336, 4}
60. {8, 1, 24, 32, 2, 384, 4}
61. {8, 4, 24, 32, 1, 384, 4}
62. {8, 5, 30, 40, 1, 480, 4}
63. {9, 1, 27, 36, 2, 432, 4}
64. {9, 4, 27, 36, 1, 432, 4}
65. {10, 1, 30, 40, 2, 480, 4}
66. {10, 4, 30, 40, 1, 480, 4}

67. {1, 10, 6, 10, 1, 150, 5}
68. {10, 1, 6, 10, 1, 150, 5}
69. {2, 5, 6, 10, 1, 150, 5}
70. {5, 2, 6, 10, 1, 150, 5}
71. {3, 5, 9, 15, 1, 225, 5}
72. {5, 3, 9, 15, 1, 225, 5}

11.16.1.5 Compiling AXMEDIS DIA Decision Taking Engine

Gecode can be compiled under Visual Studio 2005 and gcc (http://www.gecode.org/gecode-doc-latest/PageComp.html). To conform to AXMEDIS requirement of compatibility with Visual Studio 2003, the code for Decision Taking Engine dependent on Gecode libraries for constraint solving has been isolated and compiled as a Dynamic Link Libraries under Visual Studio 2005.

AxmedisCSP.dll

There are two DLLs which work hand in hand to provide a solution. All the source code for Gecode based Decision Taking Engine is contained in AxmedisCSP.dll which is loaded and used by AxmedisCSPDLL.dll.

AxmedisCSPDLL.dll

AxmedisCSPDLL.dll acquires the solution of a model from AxmedisCSP.dll as a solution vector. In turn, AxmedisCSPDLL.dll formats the solution in an appropriate manner in the form of strings and writes it to a file.

AxmedisCSPDE.exe

The AxmedisCSPDLL.dll is used by the AxmedisCSPDE executable. This process takes in as arguments the path of the file where the solution is to be written, as well as the model which is to be solved. These arguments are forwarded to the AxmedisCSPDLL.dll function solver which uses the AxmedisCSP.dll function solve to get the solution to specified model and then save the solution to a file, the path for which has been specified already.

The JSAxDecisionTakingEngine and the profile management modules i.e. JSAxUserProfile, JSAxDeviceProfile and JSAxNetworkProfile use the AxmedisCSPDE.exe process to indirectly use the Gecode DLLs.

The profile management modules are all compiled under VS 2003. They use the above mentioned executable process to request solutions of models. Any program using the AxmedisCSPDE process will need to accompany this executable with the AxmedisCSP.dll and AxmedisCSPDLL.dll files.

This approach completely separates the Gecode based constraint engine from the profile management logic, hence, avoiding the compiler incompatibility that exists between Gecode and VS2003. A few approaches were experimented with, it was learned that due to a compiler incompatibility, Gecode based DLLs of the Decision Taking Engine could not be used by the projects built under VS2003. Therefore, the approach to use files for communicating the solution with a separate process that runs Gecode based DTE was selected and has been implemented successfully.

11.16.2 Technical and Installation information

The JS class comes in the form of a static LIB. The init method has to be called in the source code where the Engine is initialized. This is necessary to add the set of functions to the Javascript Engine.

References to other major components needed	
Problems not solved	•
Configuration and execution context	

SVN repository structure

Gecode libraries and AXMEDIS Gecode based constraint engine compiled under Microsoft Visual Studio 2003 can be found on the SVN repository under:

- trunk/Applications/AxmedisConstraintEngine.

JSAx*Profile (User, Device and Network) code used with the above mentioned can be found on the SVN repository under:

- trunk/Framework/project/jsaxclasses
- trunk/Framework/include/jsaxclasses
- trunk/Framework/source/jsaxclasses
- …

Backups of previously provided code (compiled under MS VS2005) has been copied to VS2005-BAK sub folders under respective parent folders.

11.16.3 Draft User Manual

AxDecisionTakingEngine()

Constructor for Initialising the decision taking engine

Activate(String UserProfileUri, String DeviceProfileUri, String NetworkProfileUri)

Activate decision taking engine using the profiles available at the specified paths

11.16.4 Examples of usage

11.16.4.1 A Simple Example

Sample Java Script code to instantiate and then activate the decision taking engine.

```
// instantiate
var dta = new AxDecisionTakingEngine();

//activate
dta.activate(userProfile, deviceProfile, networkProfile);
```

11.16.4.2 Detailed Example: AXMEDIS AXCP DIA Adaptation Rule

A detailed example documenting the working of the AXMEDIS AXCP adaptation rule that works in conjunction with the AXMEDIS DIA Decision Engine is given below. The Decision Engine processes the AXMEDIS profiles to establish an adaptation criterion or strategy, which is in turn followed by the Adaptation rule executed in the AXMEDIS Content Processing Engine (AXCP) for adaptation of basic as well as composite AXMEDIS Objects.

Input

AXMEDIS Object:	AXOID or URI
User Profile:	URI
Device Profile:	URI
Network Profile:	URI

Output

Adapted AXMEDIS Object:	AXOID

Scripts

1. adaptSMILTemplate
2. axProfileObjectAdapter
3. axProfileObjectAdapter_Audio
4. axProfileObjectAdapter_General
5. axProfileObjectAdapter_Image
6. axProfileObjectAdapter_Text
7. axProfileObjectAdapter_Video
8. utilityFunctions
9. magickText2Image
10. miktexText2Image
11. main

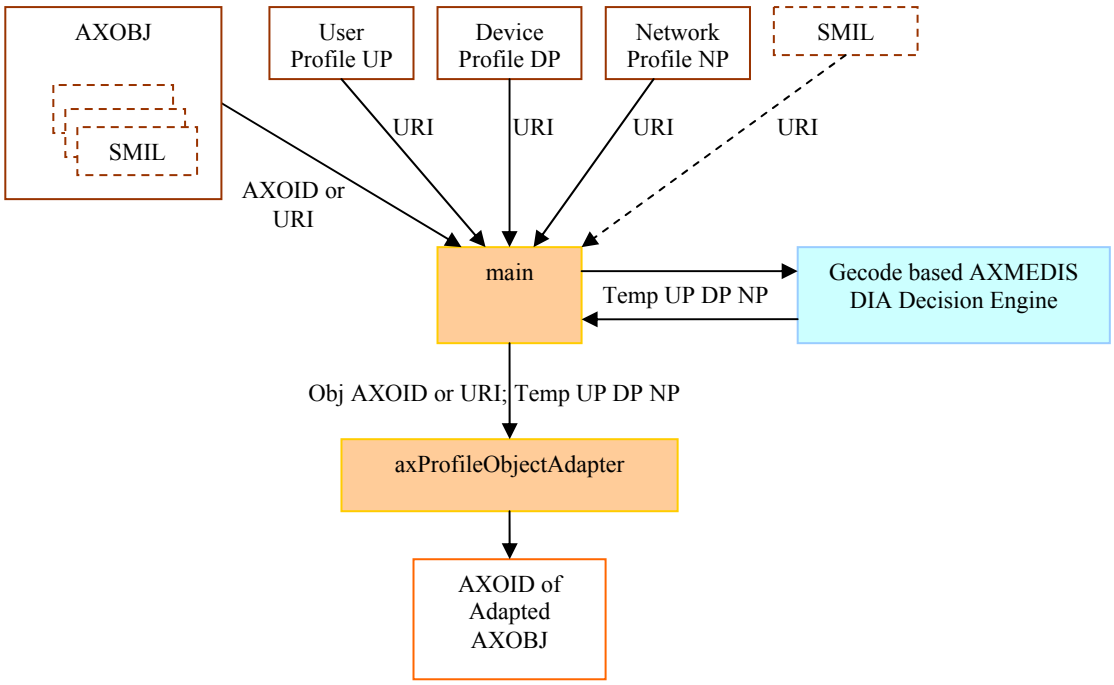
Functions

adaptSMILTemplate	
FillSmilTemplate(file)	Adapts a SMIL template (specified or extracted from the AXMEDIS Object to be adapted)
axProfileObjectAdapter	
processProfiles(user,dev,network)	Gets the elements comprising adaptation strategy from the user, device and network profiles
getRecursiveContent(obj,resArray)	Gets all types of resources from an AXMEDIS Object. In case of a composite object with multiple embedded objects, this function also returns the resources embedded within those nested/embedded objects. The returned list of resources is stored in resArray.
processSMILResources(objfilepath)	This function processes only the SMIL resources in an object located at objfilepath. It makes necessary modifications to the SMIL templates in terms of position, size, active regions etc. as well as collecting important information for resource adaptation e.g. text features, video features, region features, paramGroup features etc.
adaptObject(Resources,newAxObj)	Analyses Resources of the original object one by one. Determines type of resource and calls the relevant function for adaptation e.g. adaptAudioRes, adaptImageRes etc. Saves the returned adapted resources one by one in the newAxObj. In cases where the resource may be another AXMEDIS Object, uses a recursive call specifying Resources of that embedded object and a temporary new AxmedisObject as the two required function parameters.
axProfileObjectAdapter(AXOID,user,dev,network)	Downloads the AXMEDIS Object using AXOID/URI or loads it from localpath. Processes profiles and SMIL resources by calling relevant functions and saves the adapted SMIL resources in a new AXMEDS Object. Calls adaptObject function. Saves the new adapted object locally

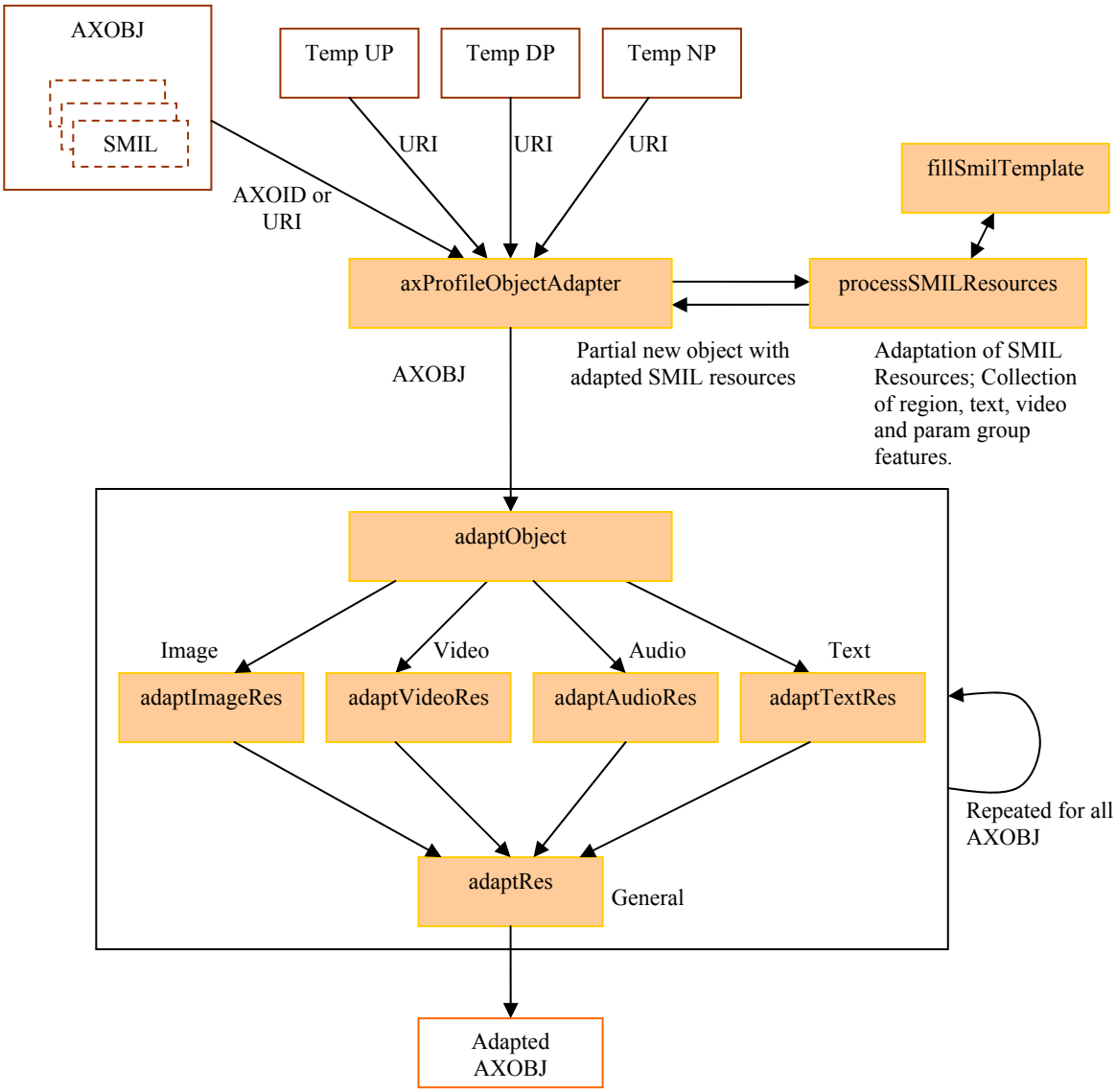
	and uploads it. Checks feasibility of transfer in light of device and network limitations.
axProfileObjectAdapter_Audio	
adaptAudioRes(res)	Adapts audio resources
axProfileObjectAdapter_General	
adaptRes(res,profInstance)	Checks contentIDs and mimeTypes of all resources to ensure accurate identification and display in AxPlayers/AxEditor. Also, keeps a check on the progressive size of adapted object.
axProfileObjectAdapter_Image	
adaptImageRes(res)	Adapts image resources
axProfileObjectAdapter_Text	
adaptTextRes(res)	Adapts text resources using MikTeX or ImageMagick for text-to-image conversion.
axProfileObjectAdapter_Video	
adaptVideoRes(res)	Adapts video resources.
utilityFunctions	
splitPath (path)	Splits a path using “\\” as delimiter
createDir (path)	Creates dir using makeDir for any depth. use "\\" as path delimiter
cloneAxmedisObjectMetadata(sourceObj,targetObj)	Copies metadata of original object to adapted object
getMimeTypeExt(res)	Gets extension from mime type
getMimeTypeFromExt(res)	Gets mime type from extension
magickText2Image	
magikTxt2Gif (imageMagikPath, tempFilePath, inputText, fontName, fontSize, bgColour, textColour, width, height)	Converts text to image using ImageMagick
miktexText2Image	
prepareTex (fileName ,string, style, color,dim)	Helps in production of batch commands that are sent to MikTeX for text-to-image conversion.
textToGif (text, tempPath, tempFile, textStyle, textColor, dimension)	Converts text to image using MikTeX.
Main	
retrieveProfileFromUri(filepath,uri)	Retrieves user, device or network profile from URI and copies it to a temporary file, which is then processed by the decision engine.
main()	Initiates the process of profile retrieval from respective URIs; if successful, sends temporary copies of profiles to the decision engine for processing and decision making, else returns error; if successful, sends the temporary profiles paths and the object URI to the axProfileObjectAdapter function to initiate the adaptation process, else returns error. Also removes the local temporary copies of the profiles if they exist.

Illustrations

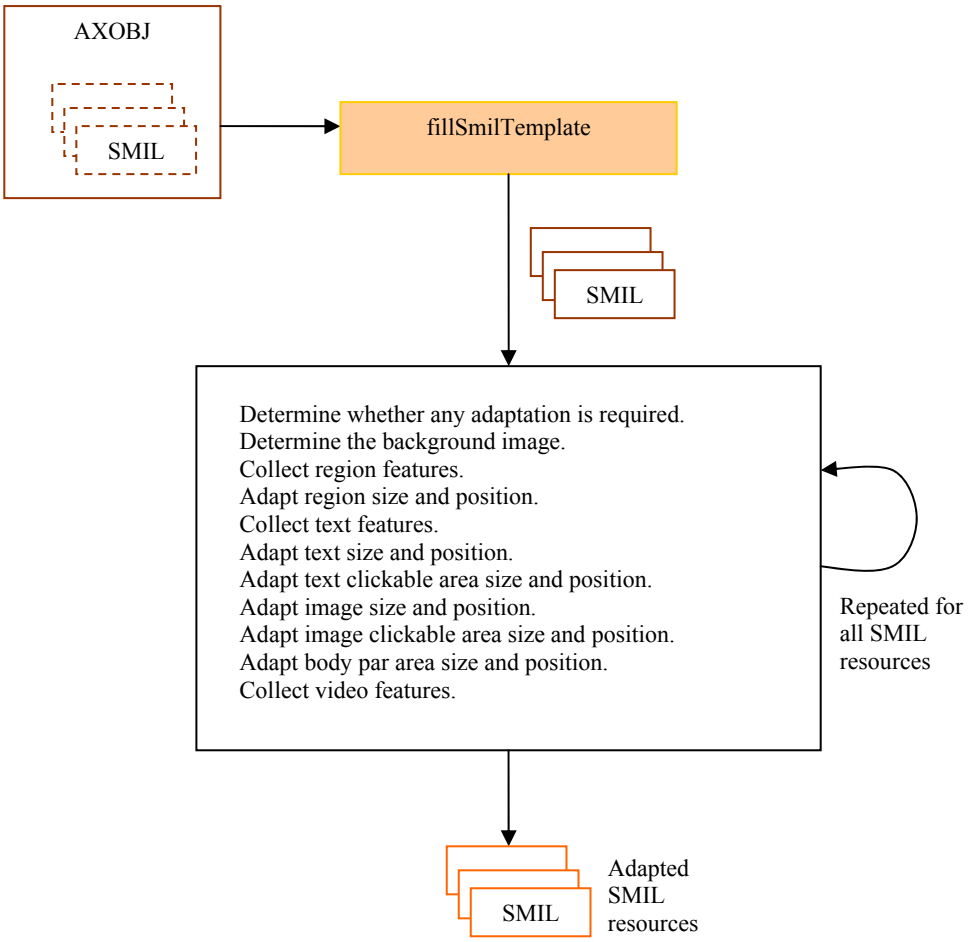
Script: main


AXOBJ
SMIL
User Profile UP
Device Profile DP
Network Profile NP
SMIL
URI
URI
URI
URI
AXOID or URI
main
Temp UP DP NP
Gecode based AXMEDIS DIA Decision Engine
Obj AXOID or URI; Temp UP DP NP
axProfileObjectAdapter
AXOID of Adapted AXOBJ

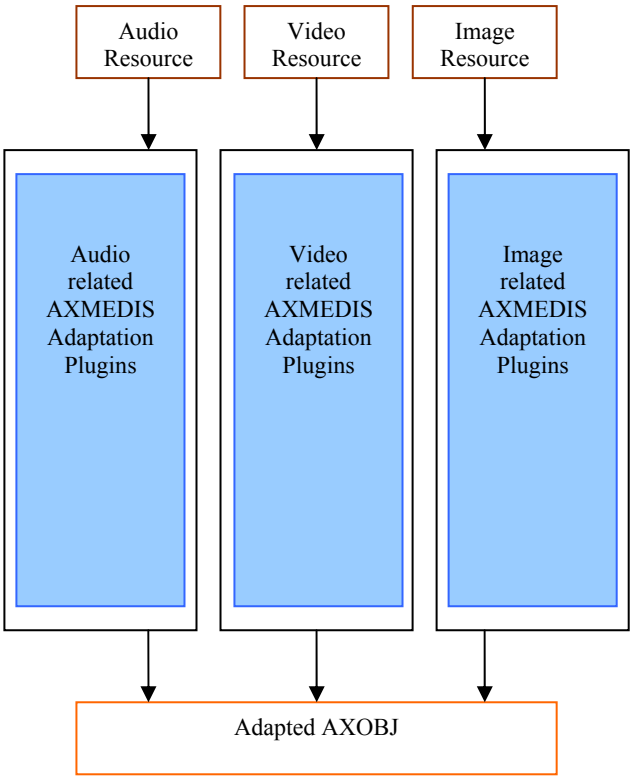
Script: axProfileObjectAdapter



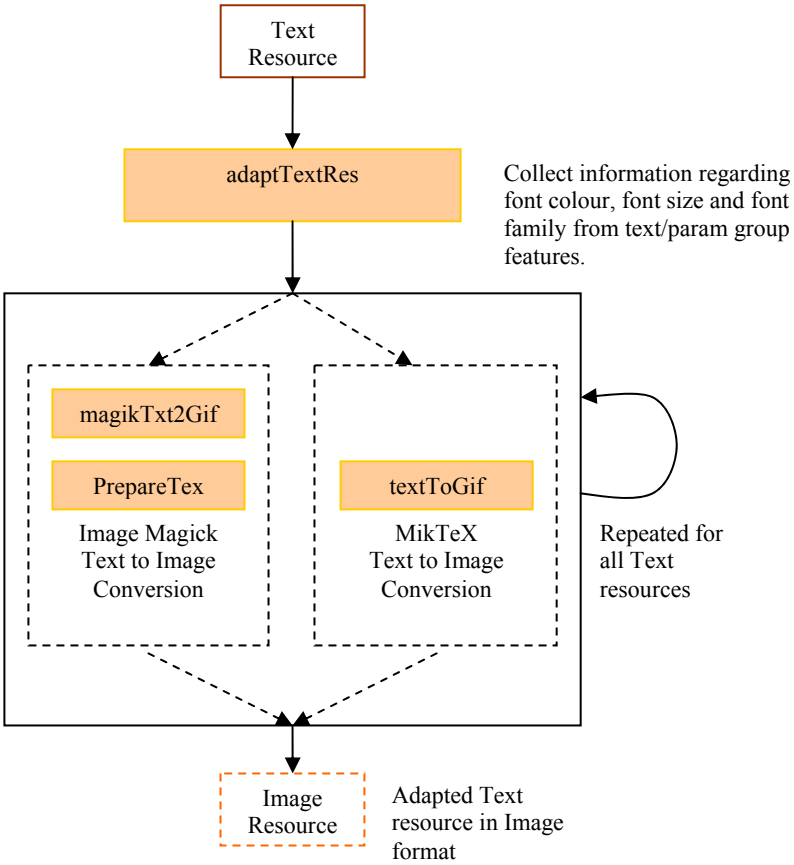
Script: adaptSmilTemplate



Script: axProfileObjectAdapter (Audio, Video and Image)



Script: axProfileObjectAdapter (Text)



11.16.5 Integration and compilation issues

None

11.17 Formatting (DSI)

Module Profile		
JSFormatting Classes		
Responsible Name	Paolo Vaccari	
Responsible Partner	DSI	
Status (proposed/approved)	approved	
Implemented/not implemented	Implemented	
Status of the implementation		
Executable or Library/module (Support)	Module	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Platforms supported	Windows, Linux, Mac OS X	
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/repos/Framework/include/jsaxclasses/ https://cvs.axmedis.org/repos/Framework/source/jsaxclasses/	
Reference to the AXFW location of the demonstrator executable tool for internal download	https://cvs.	
Reference to the AXFW location of the demonstrator executable tool for public download		
Address for accessing to WebServices if any, add accession information (user and Passwd) if any		
Test cases (present/absent)		
Test cases location	http://////////////////	
Usage of the AXMEDIS configuration manager (yes/no)	Yes	
Usage of the AXMEDIS Error Manager (yes/no)	no	
Major Problems not solved	-- --	
Major pending requirements	-- --	
Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)

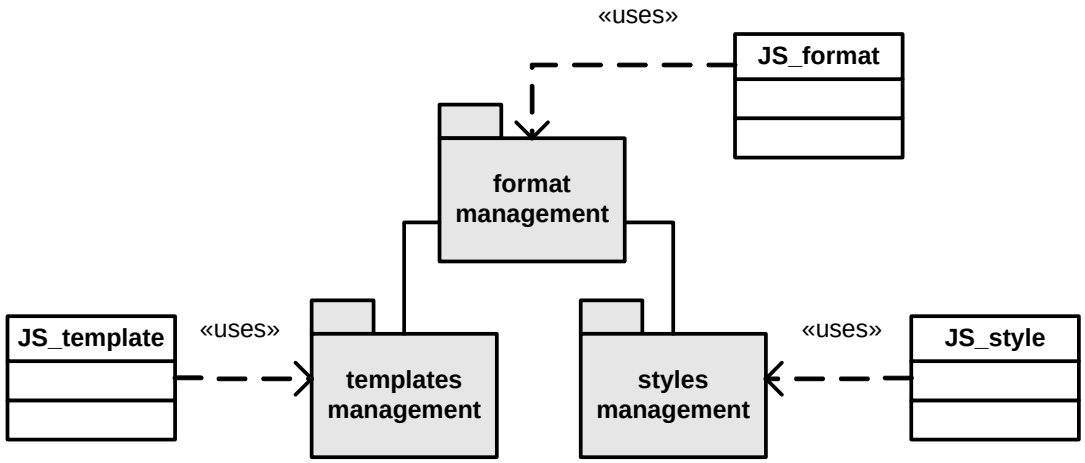
Formats Used	Shared with	format name or reference to a section
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
JS API Spidermonkey	JS API v.1.5	LGPL Licence
Xerces	Xerces-C++ v.2.7	LGPL
Xalan	Xalan-C++ v.1.10	LGPL
GALib	GALib v.2.4.6	BSD-like

The Content Format Engine functionalities and data types are accessible in JavaScript through *JS_Template*, *JS_Style* and *JS_Format* classes. They simply wrap the main classes of the C++ Format Engine:

- *JS_Template*: the interface to templates archive;
- *JS_Style*: the interface to style-sheets archive;
- *JS_Format*: the interface to selection and optimization logic, and style-sheet processing.

11.17.1 Module Design in terms of Classes

JS_Template, *JS_Style* and *JS_Format* classes expose functionalities of the C++ Format Engine.



The *JS_Format* class provides all functionalities exposed by the C++ Format_Manager to optimize style-sheets and produce the final SMIL description of the document.
JS_Format simply wraps the C++ format_manager class and exposes methods for a JavaScript Rule Executor.

The *JS_Template* class provides functionalities needed to manage the template archive.
JS_Template wraps the C++ *templates_list* class and exposes methods for a JavaScript Rule Executor.

The *JS_Style* class provides functionalities needed to manage the style-sheet archive.
JS_Style wraps the C++ *styles_list* class and exposes methods for a JavaScript Rule Executor.

11.17.2 Draft User Manual

In this section a draft description of the JSFormatting classes is provided in terms of Javascript language. For each class the set of exposed attributes, constructors and methods are reported.

JS_format
Exposed methods
addResource(String uri)
adds a new resource to the system
addResourceDescriptor(String uri)
adds a new XML resource descriptor to the system
clearResources()
empties the resources list
addCriteria(String id, String type, Int[] weights, String meta)
stores a new set of weights
setCriteria(String id)
set criteria for next filtering operation
setDevice(String uri)
setUser(String uri)
setContext(String uri)
set profiles stored in the given XML file
String[] filterTemplate()
returns a list of template IDs
String[] filterStyle(String templateId)
returns a list of style-sheet IDs
Int[] optimize(String styleId)
returns the optimized values for parameters
String apply(styleId, values)
creates the SMIL document

JS_template
Exposed methods
templates_list(String uri)
constructor; needs the address of the descriptors directory
load()
fill the list with existing template descriptors
add(String id, String location, String outformat, String category, String device, String meta)
creates a new template descriptor
String getData(String id)
String getLocation(String id)
String getMetadata(String id)
String getDevice(String id)
String getFormat(String id)
String getCategory(String id)
getter methods
String getHierarchy(String id)
returns the XML hierarchy of the resources within the template
Int getRes(String id)
returns the number of resources needed by the template
Int getResByType(String id)
returns the number of resources of the given type needed by the template
Int getType(String id)
returns the number of different resource types needed by the template

JS_style
Exposed methods
styles_list(String uri)
constructor; needs the address of the descriptors directory
load()
fill the list with existing style-sheet descriptors
add(String id, String location, String device, String metadata, String template)
creates a new style descriptor
addStyle(String xml, String id, String device, String metadata, String template)
creates and stores a new style-sheet and its descriptor
String getData(String id)
String getLocation(String id)
String getMetadata(String id)
String getDevice(String id)
String getTemplate(String id)
getter methods
Int getParamNum(String id)
returns the number of parameters defined within the style-sheet
String[] getParams(String id)
returns the list of parameters defined within the style-sheet

11.17.3 Examples of usage

Following examples illustrates usage of JS classes to perform basic formatting operations.

11.17.3.1 Loading resources of an AXMEDIS object

The *loadAxFile()* function allows loading in *rList* the resources contained into an Axmedis object (*axFile*). It calls (recursively) the function *loadResources()*, that processes all the object's childs.

```
function loadAxFile(axFile, rList)
{
```

```
    // opening file with axObject(s)
    var obj = new AxmedisObject(axFile);
    // loading resources contained in the root element
    // and starting iteration over the children
    loadResources(obj, rList);
    return true;
}

function loadResources(axObj, rList)
{
    // exploring object's hierarchy
    var children = axObj.getContent();
    var childrenCount = axObj.childrenCount;
    var resources = new Array();
    var resCount = 0;
    var objects = new Array();
    var objCount = 0;
    var child;
    for (var i = 0; i < childrenCount; i++)
    {
        child = children[i];
        if (child instanceof AxResource)
        {
            resources[resCount] = child;
            resCount++;
        }
        else // child is instanceof AxmedisObject
        {
            objects[objCount] = child;
            objCount++;
        }
    }

    // adding resources
    for (i = 0; i < resCount; i++)
    {
        var type = resources[i].mimeType;
        var path = resources[i].localPath;
        var enc = resources[i].encoding;
        var id = resources[i].contentID;
        var cat = readCategoryMetadata(axObj, id);
        var ax = axObj.AXOID;
        rList.addResource(type, path,
                          cat, axObj.AXOID, "", "metadata");
    }
    // iterating over inner objects
    for (i = 0; i < objCount; i++)
    {
        loadResources(objects[i], rList);
    }
}
```

11.17.3.2 Formatting an AXMEDIS object

The *formatObject()* function performs a complete formatting of an Axmedis object. It includes following steps:

1. loading templates and styles from system's folders;
2. loading device capabilities (and/or other preferences) from static XML profiles;
3. loading resources contained in the AxObject;
4. filtering templates and styles;
5. (optimizing style-sheet and) creating a SMIL file;
6. embedding the SMIL file into the AxObject.

The FORMATTING_CONF module of the ConfigurationManager is used to get the paths of template/style descriptors.

```
function formatObject(fm, devFile, axFile, outFile)
{
  // load templates
  fm.loadTemplateDescriptors();

  // load style-sheets
  fm.loadStyleDescriptors();

  // load profiles
  fm.loadDeviceInfo(devFile);

  // load resources
  fm.loadAxFile(axFile, fm.getResourceList());

  // filtering
  var tplIds = fm.getBestTemplates();
  var tplId = tplIds[0];
  print (" - selected template is: " + tplId);

  var stlIds = fm.getBestStyles(tplId);
  var stlId = stlIds[0];
  print (" - selected style-sheet is: " + stlId);

  // (optimize and) create SMIL file
  // fm.saveSMIL(tplId, stlId, "output.smi"); // no optimization
  print (" - optimization: please wait...");
  // var type = fm.getDeviceInfo().getDeviceClassType();
  var x = fm.getDeviceInfo().getHorizontalScreensize();
  var y = fm.getDeviceInfo().getVerticalScreensize();
  ret = fm.saveOptimizedSMIL(tplId, stlId, "output.smi", x, y);
  print (" - optimization completed");

  // include SMIL file into the object
  var obj = new AxmedisObject(axFile);
  res = new AxResource();
  res.load("output.smi");
  res.mimeType = "application/smil";
  res.contentID = "output.smi";
  res.localPath = "output.smi";
  res0 = obj.getContent()[0];
  if (res0 != null)
    obj.insertContent(res, res0, true);
  else
    obj.addContent(res);
  res.flush();
  res = null;
  obj.save(outFile);
```

```
  //obj.dispose();
  return true;
}
```

11.17.3.3 Editing resources category

The *addCategoryMetadata()* and *readCategoryMetadata()* functions allow editing of resources category. Some metadata (in XMLformat) can be embedded into AXMEDIS objects to describe category of usage of each resource

```
function addCategoryMetadata(obj, res, cat, id)
{
    var md = obj.getGenericMetadata();
    for (j = 0; j < md.length; j++)
    {
        if (md[j].getMetadataID() == id)
        {
            xmlStr = md[j].getXML();
            xmlDoc = new REXML(xmlStr);
            xmlid = xmlDoc.rootElement.childElements[0].text;
            category = xmlDoc.rootElement.childElements[1].text;
            if (category != cat)
            {
                obj.removeMetadata(md[j]);
            }
            else
                return true;
         }
    }
    var str = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
    str = str + "<axmd:resource xmlns:axmd=\"http://www.axmedis.org\">";
    str = str + "<axmd:name>";
    str = str + res;
    str = str + "</axmd:name>";
    str = str + "<axmd:category>";
    str = str + cat;
    str = str + "</axmd:category>";
    str = str + "</axmd:resource>";
    var axmd = new AxMetadata(str);
    axmd.setMetadataID(id);
    obj.addMetadata(axmd);
    return true;
 }

function readCategoryMetadata(obj, id)
 {
    var md = obj.getGenericMetadata();
    for (j = 0; j < md.length; j++)
    {
        if (md[j].getMetadataID() == id)
        {
            xmlStr = md[j].getXML();
            xmlDoc = new REXML(xmlStr);
            xmlid = xmlDoc.rootElement.childElements[0].text;
            category = xmlDoc.rootElement.childElements[1].text;
```

```
            return category;
        }
    }
    return null;
}
```

In the following example a set of resources is extracted from an AXMEDIS object and a SMIL document is produced to integrate the resources in a multimedia presentation; finally, the SMIL document is included in the AXMEDIS object.

11.18 JS_MetadataMapper (UNIVLEEDS)

Module Profile	
JS_MetadataMapper Class	
Responsible Name	Garry Quested
Responsible Partner	UNIVLEEDS
Status (proposed/approved)	approved
Implemented/not implemented	Not Implemented
Status of the implementation	
Executable or Library/module (Support)	Module
Single Thread or Multithread	Multithread
Language of Development	C++
Platforms supported	Windows
Reference to the AXFW location of the source code demonstrator	https://cvs.axmedis.org/newrepos/Framework/include/jsaxclasses/metadatamapper https://cvs.axmedis.org/newrepos/Framework/source/jsaxclasses/metadatamapper
Reference to the AXFW location of the demonstrator executable tool for internal download	
Reference to the AXFW location of the demonstrator executable tool for public download	
Address for accessing to WebServices if any, add accession information (user and Passwd) if any	
Test cases (present/absent)	
Test cases location	https://cvs.axmedis.org/newrepos/Framework/doc/test/jsaxclasses/jsmetadatamappertest
Usage of the AXMEDIS configuration manager (yes/no)	no
Usage of the AXMEDIS Error Manager (yes/no)	no
Major Problems not solved	-- --
Major pending requirements	-- --

Interfaces API with other tools, named as	Name of the communicating tools References to other major components needed	Communication model and format (protected or not, etc.)
Formats Used	Shared with	format name or reference to a section
Yes		XSLT, Standard format
Protocol Used	Shared with	Protocol name or reference to a section
Used Database name		
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	Xerces C++ Parser v-2.7.0	Apache Software Licence, v-2.0
Xalan	Xalan-c++ v-1.9	The Apache Software License, v1.1
wxWidgets	wxWidgets-2.4.2	wxWindows licence (http://www.wxwidgets.org/newlicen.htm)

JS_MetadataMapper is a JavaScript wrapper around the MetadataMapper C++ library. It provides the following functionalities:

1) Instantiating a MetadataMapper object within a JavaScript
2) Loading an XSLT stylesheet
3) Transforming XML from source to destination language according to the stylesheet

11.18.1 Module Design in terms of Classes

JSMDM Class

11.18.2 Draft User Manual

This is a draft User Manual for the JS classes.

JSMDM

Class name in Javascript = *MetaDataMapper*

Exposed methods

MetaDataMapper()

It creates *a mapper* object

LoadXSLFile(string xsltFilePath)

load an XSLT file into a DOMDocument

LoadXSL(string xsltXMLString)

load an XSLT string into a DOMDocument

TransformFile(string infile, string outfile)

transform metadata in input file according to stylesheet to create an output file

Transform(string inxml, string outxml)
transform metadata in an input string according to stylesheet to create an output string

11.18.3 Examples of usage

```
// Example 1: create a new file containing transformed metadata
var mdm = new MetaDataMapper();
mdm.LoadXSLFile("style.xsl");
mdm.TransformFile("input.xml","output.xml");

// Example 2: create a new file containing transformed metadata
var input = "<?xml ….."; // input xml string
style = "<?xml ….."; // stylesheet xml string
var output;// output xml string
var mdm2 = new MetaDataMapper();
mdm2.LoadXSL(style);
mdm2.Transform(input,output);
```

11.18.4 Technical and Installation information

The JS Class comes in the form of a static LIB.

References to other major components needed	
Problems not solved	
Configuration and execution context	It is used within the AXCP Rule Editor and Engine.

Information about the installation of JS modules within the AXCP Rule Editor and Engine can be found in section "AXMEDIS DATA Types and Functions for JavaScript".