Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

AXMEDIS Content Processing GRID: Script Language User Manuals

(addition to DE5-0-1-1 AXFW User Manual)

Version: 1.2
Date: 28/03/2007
Responsible: DSI: I. Bruno, N. Mitolo (revised and approved by coordinator)

Project Number: IST-2-511299 Project Title: AXMEDIS Deliverable Type: report Visible to User Groups: yes Visible to Affiliated: yes Visible to the Public: yes
Deliverable Number: additional annex to DE5.0.1.1 Contractual Date of Delivery: M30 Actual Date of Delivery: 28/03/2006 Title of Deliverable: AXMEDIS content processing GRID, Script Language User Manual Work-Package contributing to the Deliverable: WP5 Task contributing to the Deliverable: WP5 Nature of the Deliverable: report Author(s): DSI and all partners involved in the AXCP content processing

Abstract: This document contains the user manuals of the major tools provided by AXMEDIS

Keyword List: MPEG-21, AXOM, transcoding, adaptation, video, image, document, ringtones, profiling, formatting, styling, SMIL, XSLT, metadata, mapping, GRID control.

Table of Content

1 INTRODUCTION 6

2 AXMEDIS OVERVIEW OF CONTENT PROCESSING CAPABILITIES 6

3 AXMEDIS FRAMEWORK 9

3.1 ACCESSING TO THE AXMEDIS FRAMEWORK 10
3.2 AXMEDIS CONTENT PRODUCTION 11
3.3 AXMEDIS CONTENT PROCESSING CAPABILITIES 12
3.4 AXMEDIS CONTENT PROCESSING TOOLS 13
3.5 AXMEDIS EDITORS, THE AUTHORING TOOLS 14
3.6 CONTENT ADAPTATION FACILITIES 15
3.7 CONTENT FINGERPRINT AND DESCRIPTORS EXTRACTION 16

***4* AXMEDIS CONTENT PROCESSING SCRIPT LANGUAGE USER MANUAL 18**

4.1 AXOM, AXMEDIS MODEL AND JAVASCRIPT CLASSESS 18
4.1.1 AxmedisObject 18
4.1.2 AxInfo 20
4.1.3 AxResource 28
4.1.4 AxDublinCore 28
4.1.5 AxMetadata 29
4.1.6 Examples of usage 29
4.2 NETWORK 31
4.2.1 Form 31
4.2.2 HttpConnection 31
4.2.3 FtpConnection 32
4.2.4 OdbcConnection 33
4.2.5 WebServiceConnection 33
4.2.6 Examples of usage 34
4.3 SEARCHBOX JAVASCRIPT CLASSES 34
4.3.1 AXSearchbox 35
4.3.2 Document 36
4.3.3 MetadataValue 36
4.3.4 DocumentMetadata 36
4.3.5 QueryParser 37
4.3.6 QueryInfo 37
4.3.7 QueryView 37
4.3.8 QuerySort 37
4.3.9 QueryAtomType 37
4.3.10 QueryAtom 38
4.3.11 QuerySliceWeight 38
4.3.12 QuerySpec 39
4.3.13 QueryResult 40
4.3.14 Examples of usage 40
4.4 LICENSE AND JAVASCRIPT CLASSES 41
4.4.1 Condition 42
4.4.2 Fee 42
4.4.3 Grant 43
4.4.4 Principal 44
4.4.5 GrantGroup 44
4.4.6 Interval 44
4.4.7 Issuer 44
4.4.8 License 45
4.4.9 LicenseManager 46
4.4.10 LicensePattern 46
4.4.11 OMALicense 47

4.4.12 Number 47
4.4.13 Resource 47
4.4.14 Right 48
4.4.15 Territory 48
4.4.16 PAR 48
4.4.17 PARGrant 49
4.4.18 PARGrantGroup 49
4.4.19 PARResource 50
4.4.20 PARRight 50
4.4.21 PMSClient 50
4.4.22 Contract 56
4.4.23 RightseExpressionTranslator 56
4.4.24 RightsParameters 56
4.4.25 RightsFeeParameters 58
4.4.26 RightsIntervalParameters 58
4.4.27 RightsLimitParameters 59
4.4.28 RightsRegionParameters 59
4.4.29 PARGrantParameters 60
4.4.30 AXToolInfoParameters 60
4.4.31 ProtInfoParameters 61
4.4.32 OpInfoParameters 62
4.4.33 AXIDParameters 63
4.5 SELECTION JAVASCRIPT CLASSES 64
4.5.1 Class Name 64
4.5.2 Examples of usage 65
4.6 FORMATTING ENGINE FUNCTIONS 65
4.6.1 FormatManager 65
4.6.2 ResourceList 66
4.6.3 DeviceInfo 66
4.6.4 TemplateList 67
4.6.5 StyleList 67
4.6.6 Examples of usage 67
4.7 PROFILING JAVASCRIPT CLASSES 69
4.7.1 JSAxUserProfile 69
4.7.2 JSAxDeviceProfile 70
4.7.3 JSAxNetworkProfile 71
4.7.4 Examples of usage 72
4.8 METADATA MAPPER JAVASCRIPT CLASS 72
4.8.1 MetaDataMapper 72
4.8.2 Examples of usage 73
4.9 AXEPTOOL JAVASCRIPT CLASSES 73
4.9.1 AXP2PManager 73
4.9.2 DownloadedObjectListResponse 75
4.9.3 DownloadStatusResponse 75
4.9.4 PublicationStatusResponse 76
4.9.5 PublishedObjectListResponse 76
4.9.6 Examples of usage 77
4.10 AXMEDIS JAVASCRIPT FUNCTIONS 77
4.10.1 I/O functions 77
4.10.2 File functions 77
4.10.3 Dir functions 78
4.10.4 Process Functions 78
4.10.5 Mime Type Functions 78
4.10.6 Examples of usage 78
4.11 UTILITY AND OTHER JAVASCRIPT CLASSES 79
4.11.1 ZipArchiver 79
4.11.2 Examples of usage 80

5 E4X SHORT GUIDE FOR JAVASCRIPT V. 1.6 81

5.1 KNOWN BUGS AND LIMITATIONS 81
5.2 RESOURCES 81

5.3 E4X - KEYWORDS, STATEMENTS, IDENTIFIERS AND PUNCTUATORS 81
5.3.1 Context Keywords 81
5.3.2 Statements 81
5.3.3 Identifiers and Punctuators 82
5.4 E4X - OPERATORS & OBJECTS 84
5.4.1 Operators 84
5.4.2 E4X Objects 85
5.5 E4X - GLOBAL METHODS AND PROPERTIES 91
5.5.1 Global Methods and Properties 91
5.5.2 Advantages and Limitations of E4X 93
5.5.3 E4X by Example 93

6 AXMEDIS CONTENT PROCESSING FUNCTIONALITIES IN PLUG-INS 95

6.1 AUDIO ADAPTATION PLUG-IN 95
6.1.1 FFAudioTranscoding 95
6.1.2 LSAudioTranscoding 96
6.2 AUDIO DESCRIPTOR PLUG-IN 98
6.2.1 LowLevelDescriptors 98
6.2.2 Segmentation 100
6.2.3 TempoEstimation 100
6.2.4 MusicGenreEstimation 101
6.3 AUDIO FINGERPRINT EXTRACTION PLUG-IN 103
6.3.1 AxAFPExtract 103
6.3.2 AxAFPCompare 104
6.4 TEXT DESCRIPTORS PLUG-IN 106
6.4.1 KWFromComparisons 106
6.4.2 KWFromSemanticAnalysis 106
6.5 TEXT TO DOCS ADAPTATION PLUG-IN 108
6.5.1 DocumentConversion 108
6.6 TEXTFINGERPRINT 109
6.6.1 MD5Hash 109
6.7 MULTIMEDIA ADAPTATION PLUG-IN 110
6.7.1 Mp4To3gp 110
6.7.2 Mp4ToISMA 111
6.7.3 AddMultimediaFiles 111
6.7.4 ToMp4 113
6.7.5 ExtractMediaTrack 113
6.7.6 CatMultimediaFiles 114
6.7.7 DelayTrack 115
6.7.8 RemoveTrack 115
6.7.9 ExtractFromStartToEnd 116
6.7.10 Mp4ToAvi 117
6.8 RINGTONE ADAPTATION PLUG-IN 119
6.8.1 convert 119
6.8.2 convert_to_MP3 119
6.8.3 convert_to_WAV 120
6.8.4 resample 120
6.8.5 convert and resample 121
6.8.6 getInfo 122
6.8.7 clip 123
6.9 AUDIO RECOGNITION AND MONITORING PLUGIN 125
6.10 IMAGE PROCESSING PLUGIN 125
6.10.1 Main functionalities 125
6.10.2 Relationship with other tools 126
6.10.3 Detailed description of the functionalities and Screenshots 126
6.10.3.1 Conversion 127
6.10.3.2 Import 128
6.10.3.3 Resize 128
6.10.3.4 Contrast 129
6.10.3.5 Edge 130
6.10.3.6 Emboss 130
6.10.3.7 Blur 131

6.10.3.8 GaussianBlur 131
6.10.3.9 Median 132
6.10.3.10 Mirror 133
6.10.3.11 Noise 133
6.10.3.12 Despeckle 134
6.10.3.13 Equalize 134
6.10.3.14 Enhance 136
6.10.3.15 ExtractChannel 136
6.10.3.16 Grayscale 137
6.10.3.17 Magnify 137
6.10.3.18 Minify 138
6.10.3.19 Modulate 138
6.10.3.20 Monochrome 139
6.10.3.21 Negate 139
6.10.3.22 Normalize 140
6.10.3.23 OilPaint 140
6.10.3.24 Quality 141
6.10.3.25 Quantize 141
6.10.3.26 Raise 142
6.10.3.27 ReduceNoise 143
6.10.3.28 Replace 143
6.10.3.29 FloodFill 144
6.10.3.30 Roll 145
6.10.3.31 Rotate 146
6.10.3.32 Scale 147
6.10.3.33 Shear 147
6.10.3.34 Shade 148
6.10.3.35 Spread 149
6.10.3.36 SetOpacity 149
6.10.3.37 SubImage 150
6.10.3.38 GetInfo 151
6.10.3.39 SetMaskColour 151
6.10.3.40 Paste 152
6.10.3.41 Test 153
6.11 VIDEOADAPTATION 154
6.11.1 AX_ffmpegTranscoder 154
6.12 VIDEOFINGERPRINTEXTRACTION 157
6.12.1 AxVFPExtract 157
6.12.2 AxVFPCompare 157
6.13 PLAGIARISM 159
6.13.1 Compare 159
6.14 LANGUAGEGUESSER 160
6.14.1 LanguageGuesser 160

1 Introduction

This deliverable aim is to describe the AXMEDIS Script included in the AXMEDIS tools.

This document should be consulted together with:

- AXMEDIS Framework for ALL: axmedis-de5-1-2-1-axmedis-for-all-v2-7.pdf , http://www.axmedis.org/documenti/view_documenti.php?doc_id=1728
- AXMEDIS Major Tools User Manual: axmedis-de5-0-1-1-axmedis-user-manuals-v-1-2.pdf, http://www.axmedis.org/documenti/view_documenti.php?doc_id=2347
- AXMEDIS Framework Guidelines updated version with AXMEDIS Plug in SDK axmedis-de5-1-3-1-framework-guidelines-v2-2.pdf, http://www.axmedis.org/documenti/view_documenti.php?doc_id=2197

2 AXMEDIS overview of content processing capabilities

A content factory can be built on the basis of AXMEDIS tools in a scalable and flexible manner. Also tuning for example, GRID size, database size and type, number of authoring tools, number and types of tools/algorithms and libraries for processing content, licenses, integration support based on Workflow or not, etc. This allows setting up a large range of configurations to satisfy the needs of small and large content producers, integrators, and distributors.

The **AXMEDIS Database Area** includes the AXMEDIS/MPEG-21 database model, supporting the storage and access to AXMEDIS content via a large set of metadata for each object grouped in what it is called AXInfo, and that can be customized with your needs. The database also includes produced licenses for the objects, history of performed actions on content, potentially available rights for each digital resource, models of contracts, etc. The AXInfo includes Dublin core plus descriptors and many other metadata for managing protection, lifecycle, etc. Any descriptors and metadata can be added in a flexible manner. Thus, different AXMEDIS factories may be based on different AXInfo and metadata, while automatic adapters can be defined and activated. The database area is based on a scalable database, a powerful AXMEDIS Database manager, and an effective **AXMEDIS Query Support** endowed of an easy to use user interface. The User may perform queries to search for objects and content located in the CMSs, in the local AXMEDIS database and in the virtual database comprised of the AXMEDIS content accessible/published via the P2P network of AXEPTools in the AXMEDIS B2B Network.

The **AXMEDIS Content Processing Area** (AXCP Area) is based on a GRID solution for automating all the activities to be performed for the production, and processing of content. The major tools are the **AXCP GRID Node (Engine) and AXCP Scheduler,** which are respectively the single node (computer) of the GRID and the organizer of processes on the GRID Nodes. They implement a scalable solution to process from smaller collections to huge amount of content per day, per minute. The processing algorithms can be specified in terms of script code (in Spider Monkey) allowing the manipulation of complex AXMEDIS data types and simple digital resources and content in general, and for the direct access to the AXMEDIS database and processing queries with the help of the AXMEDIS Query Support. The solution allows the writing of any kind of content processing algorithms, to activate them automatically on some query result, and these can be put in execution as independent processes on a scalable GRID for massive production and processing of digital resources in respect of the DRM.

The available data types, operators and accessible algorithms allow manipulation of any digital resources in a large number of formats. Algorithms can be defined for massive content composition (packaging, combination, etc.) and content layout formatting (synchronization, image and screen layout, from image sequence to video, etc.), content adaptation (change in resolution, subsampling, change in format, etc.), transcoding, coding, decoding, fingerprint extraction, estimation of descriptors, license adaptation and transcoding, license production and verification, etc.;

The users of the **AXMEDIS Content Processing Area** can code in terms of Java Script rules any kind of processing procedures and algorithms to manipulate/produce:

- Any digital resource:
    - such as images (more than 150 different formats), audio (more than 50 formats), video (more than 50 formats), documents (TXT, PS, HTML, PDF, RTF, DOC, etc.), multimedia (more than 20 formats including MPEG-4, HTML, LOM, etc.);
    - for their transcoding, adaptation, feature and descriptor extraction, recognition, certification, etc.;
    - with functionalities of many well know and powerful processing libraries such as: FFMPEG lib, LibSNDFile, TreTagger, DocFrac, GhostScript, XPDF, HTMLDOC, ImageMagik, MP4Box, Xerces, XALAN, CCPP, etc. (if you are interested in adding more libraries please contact AXMEDIS people);
- Packages and their composition and formatting
    - AXMEDIS objects with AXInfo Metadata and indexing,
    - MPEG-21 Objects,
    - including digital resources, metadata (e.g., Dublin Core, etc.),
    - protection information, etc.
- Protected objects and resources, managing protection information:
    - by using MPEG-21 IPMP model, and format
    - using algorithms such as encryption/decryption, scramble, compression, key production, different sizes for keys, etc.
- Licenses on the basis of the business models chosen:
    - stating grants, conditions, etc.,
    - verifying license consistency with respect to the potentially available rights, with the license in production, etc.,
    - by using formalism of MPEG-21 REL, and with OMA ODRL – MPEG-21 REL transcoding
- Automatic content and information access
    - database accesses (ODBC, etc.) with direct facilities;
    - database access by means of crawling facilities to access to a larger set of possible database models. They may contain digital content, resources, files, metadata, administrative and licensing information, etc., and can be physically located in several different computer systems and based on several different database models: ODBC, MySQL, ORACLE, MS-SQL, Tamino Lobster XML, etc., or files systems. The access to this information is performed by means of Focuseek Crawler;
    - file system and operating system access;
    - http and ftp accesses;
    - AXMEDIS database access with query support, actualization of selections, active queries, etc.
- Device capabilities format and processing facilities, to take into account for adaptation and/or processing;
- User Profile format and processing facilities, to take into account for adaptation and/or processing;
- WSDL facilities for the activation of WEB services dynamically on the basis of their definition;
- XML facilities for the application of styles and general processing;
- SMIL facilities for the application of templates and styles and processing;
- etc.

The algorithms and procedures used in the AXCP Area can be expanded by using the AXMEDIS Plug In technology that allows customizing and easily expanding the processing capabilities of the AXMEDIS GRID. Algorithms for the extraction of fingerprint, descriptors, adaptation, content processing, DRM adaptation, metadata adaptation, are built as pluggable algorithms. ***Any other library, model and format and related algorithms for their manipulation can be plugged in the AXCP in a very easy manner.***

The AXMEDIS **Workflow Management tools** include a set of micro tools and interfaces which are pervasively connected to all the AXMEDIS tools and plug-ins to allow interfacing the whole content factory to Workflow tools such as Open Flow and BizTalk. The control is performed to define AXMEDIS factory workflow policies and to manage inter-factory workflows policies.

The **AXMEDIS Editor** is the authoring tool for manually producing AXMEDIS objects when needed and for supporting the designer to create the scripts for the AXCP that could be considered macros of the AXMEDIS Editor. It is based on the AXMEDIS Object Model, called AXOM and based on MPEG-21, and all the modules and tools to manipulate and create AXMEDIS objects and related information and digital resources such as:

- resource hierarchy viewer and editor;
- visual and behavioral viewer and editor to show/manipulate visual and temporal aspects of related digital resources according to SMIL;
- metadata editor and viewer, to manipulate and view general XML metadata and specific AXInfo metadata;
- DRM viewer and editor to create and verify the licenses;
- Protection Information viewer and editor to specify, apply and browse protection aspects on the basis of the MPEG-21 IPMP format with extension of AXMEDIS;
- set of plug-ins to use algorithms for content processing as those mentioned and used in the AXCP Area mentioned above;
- set of plug-ins to allow the integration of AXMEDIS Editor within other editing and viewing applications such as: Video Editors, Image Editors, etc.;
- an interface to connect the AXMEDIS Editor with other external powerful editor tools;
- an interface with workflow (OpenFlow and BizTalk);
- set of internal viewers and players for digital resources such as document, images, video, MPEG-4, and audio resources, etc., for more than 250 different file formats.

3 AXMEDIS Framework

The AXMEDIS Framework is the set of information and tools that is at the basis of the above mentioned applications and solutions. In the next Figure, the simplified version of the AXMEDIS Framework structure is reported. It contains all the necessary tools to set up a large set of services and solutions in the area of content production, protection and distribution. The AXMEDIS Framework is an infrastructure on which several other models for content modeling, protection, production, DRM and distribution can be built in a very simple manner reusing the components and functionalities provided.

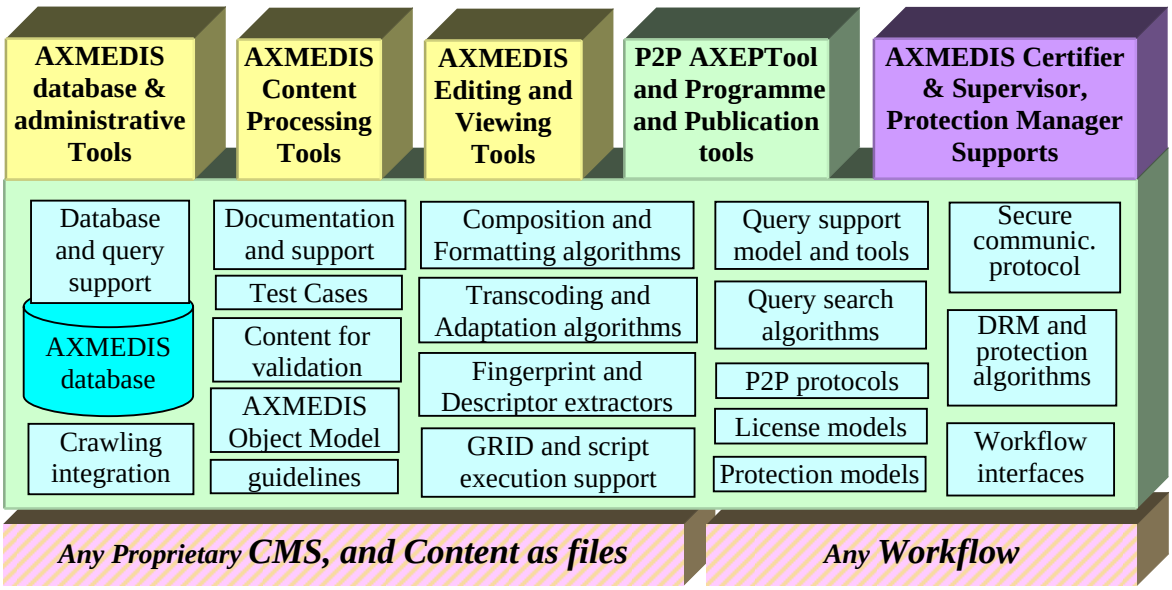


Fig.4 -- AXMEDIS Framework structure, a simplified view

The general infrastructure gives a common ground on the base of which other content based applications and tools can be built. In addition, to the modules and tools described before, the most relevant parts of the **AXMEDIS Framework** are:

- Requirements and their revisions,
- test cases and uses cases and their revisions,
- content for validations, both single resources and metadata and demonstrative AXMEDIS objects,
- general documentation of AXMEDIS tools and supports, including the:
  - whole specification of the AXMEDIS framework and the
  - detailed technical documentation of the source code,
- CVS tree with source codes of the several modules of the AXMEDIS framework,
- examples of AXCP scripts modeling algorithms for content compositions and formatting, for transcoding and adaptation, for extraction of fingerprint and descriptors, content processing, license manipulation and verification, license adaptation, etc., for many different formats of digital resources and for any categories of them: audio, video, document, multimedia, images, animations, text, metadata, etc.,
- examples and models of licenses,
- example and models for protection information,
- examples of workflow usage and programming for controlling AXMEDIS Factories,
- examples of queries and selections for accessing to the database,
- tutorials on content:
  - general aspects and state of the art,
  - content production,
  - content protection,
  - on AXMEDIS tools,

    - on distribution tools,
    - on general AXMEDIS aspects, etc.,
- guidelines for source code production for contributing to the AXMEDIS framework,
- guidelines on content production and distribution,
- guidelines for the production of AXMEDIS Plug-ins for AXCP and AXMEDIS Editors,
- guidelines for the production of licenses on the basis of contracts,
- ready to use/install AXMEDIS tools such as: AXMEDIS Players, AXEPTool, AXMEDIA tool, AXMEDIS Editors, AXMEDIS Programme and Publication tools, AXMEDIS Content Processing Tools, AXCS, AXMEDIS PMS.
-

3.1 Accessing to the AXMEDIS Framework

The present status of the AXMEDIS Framework can be obtained from its coordinator or partners. Demonstrations of the AXMEDIS tools and of the whole AXMEDIS Framework are provided at AXMEDIS conferences and in other occasions listed on the AXMEDIS Portal. The AXMEDIS Framework can be accessed by all affiliated partners. The Affiliation to AXMEDIS is performed by subscribing an Affiliation Agreement with an AXMEDIS Contractor. The Affiliation Agreement and the list of Contractors are accessible on the AXMEDIS portal.

There are many **reasons to get affiliated to AXMEDIS,** which can be summarized as follow:

- Obtaining access to an *open platform* that can be customized for your production., protection and distribution needs;
- *Reduction of costs* for content gathering, processing, production, protection and distribution;
- Adopting a standard model (MPEG-21) for content and licenses modeling and thus for inserting DRM in your business;
- Establishing contacts with other business partners interested in exploiting similar technology;
- Acquiring a greater control about content usage;
- Creating customized players;
- Exploiting and trial of new business models;
- Exploiting capabilities of secure legal P2P distribution;
- Setting up and create a customized distribution channel interoperable with others;
- Setting up some new service (empowering your present solution) on the basis of AXMEDIS technology;
- Setting up of one-stop service for content protection and DRM set up;
- Allowing reporting to your business customers which rights are exploited on their content;
- Allowing the management of rights reporting for multimedia products;
- Allowing using a solution that can be safer and more flexible with respect to state of the art;
- Saving money in accessing at innovative technologies for content production and distribution, integrated environment;
- Accessing to strongly innovative technology to trial it;
- Contributing to the AXMEDIS Framework is allowing you to continuing accessing to the framework reducing the costs for its accessibility.

Research institutions and technology providers are interested in getting affiliated with AXMEDIS to:

- make visible, promote, produced algorithms and tools that can be used for content processing and modeling and that can be in some how integrated into the AXFW. These tools may be provided as demonstrators with limited capabilities;
- exploit the AXMEDIS Framework to make business with it for the reasons reported in the above list;
- add new content models and new DRM models and make them interoperable with MPEG-21 and others already in place on AXMEDIS;
- test new algorithms and tools with respect to the state of the art solutions, in a very easy and cheep manner;
- access at low cost a framework by means of which several different configurations and solutions may be built to cover the needs of the value chain actors and tested with low effort;

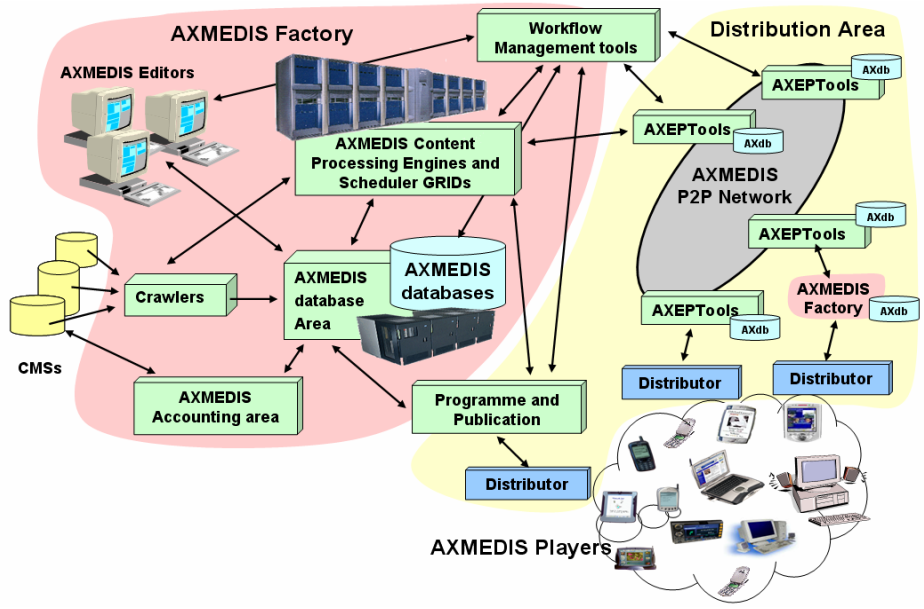
- access at tools based on MPEG-21 standard;
- collaborate with very relevant and well known research institution and companies of the areas;
- etc.

The present **status of the AXMEDIS Framework** can be obtained from its coordinator. Demonstrations of the AXMEDIS tools and of the whole AXMEDIS Framework are provided at AXMEDIS conferences and in other occasions listed on the AXMEDIS Portal. The AXMEDIS Framework can be accessed by Affiliated Partners. The Affiliation to AXMEDIS may be performed by subscribing an Affiliation Agreement with an AXMEDIS Contractor.

The AXMEDIS 2006 conference will be held in Leeds in December 2006. The Call For Papers is open until April 2006. http://www.axmedis.org/axmedis2006/

3.2 AXMEDIS Content Production

The Content Production area of AXMEDIS is mainly focussed on what it is called AXMEDIS Factory automating: (i) the packaging containing the digital resources (the real content), (ii) adapting and transcoding, (iii) protecting content and producing corresponding Prot-Info, (iv) publishing and distributing the produced package, (v) producing licenses for the users, etc.



Content Object Production

- Existing contents (e.g., resources and metadata located in databases, CMSs, file systems) are crawled and collected using automated processes and rules (e.g., Crawler, rule editing);
- Crawled contents or new contents can be inserted into the AXMEDIS Database;
- Content can be automatically packaged in AXMEDIS model (which is MPEG-21 compliant);
- Content production and elements (metadata, resources, protection information and licenses) can be processed manually with authoring tools and editor or automatically with AXMEDIS Content Processing tools based on executable rules that can be hosted on single or massive GRID of computers;
- Authoring tools can be used to insert/revise metadata, define protection information, define DRM licenses models, to modify content or simply to view the objects;
- Various processing are offered (adaptation, transcoding, protection, etc.), either automatically or manually (using GUI editor);
- The various components and digital resources can be glued together by means of SMIL based templates and style that may be used to define the usage interface (format) of the whole object: karaoke, collection, menus, sliding presentation, buttons, live, animations, etc.;

- Results can be sorted in a database or on file systems or published in towards distributors or on the B2B P2P AXMEDIS network;
- They are now available on the AXMEDIS network, for further aggregation, distribution, etc., to be searched, modified, or shared;
- Queries and P2P allow retrieving content located in all the connected AXMEDIS Factories;
- The queries can be activated to automatically react at eventual changes in the sources, and thus to perform an automatic production/update;
- Queries can be performed on the basis of classification and identification metadata, but also on technical features, descriptors, licensing information (PAR), etc.;
- Once identified the objects or queries, they can be used as input parameter of processing rules for the GRID AXMEDIS Content Processing;
- All these activities can be governed by Workflow Management Tools for defining process production flow and information of the content factory and among different factories.

3.3 AXMEDIS Content Processing Capabilities

The AXMEDIS Content Processing Area has been designed to provide a set of digital content processing tools to aid content designers to create rules/script for **automating** content production and processing and in more details for:

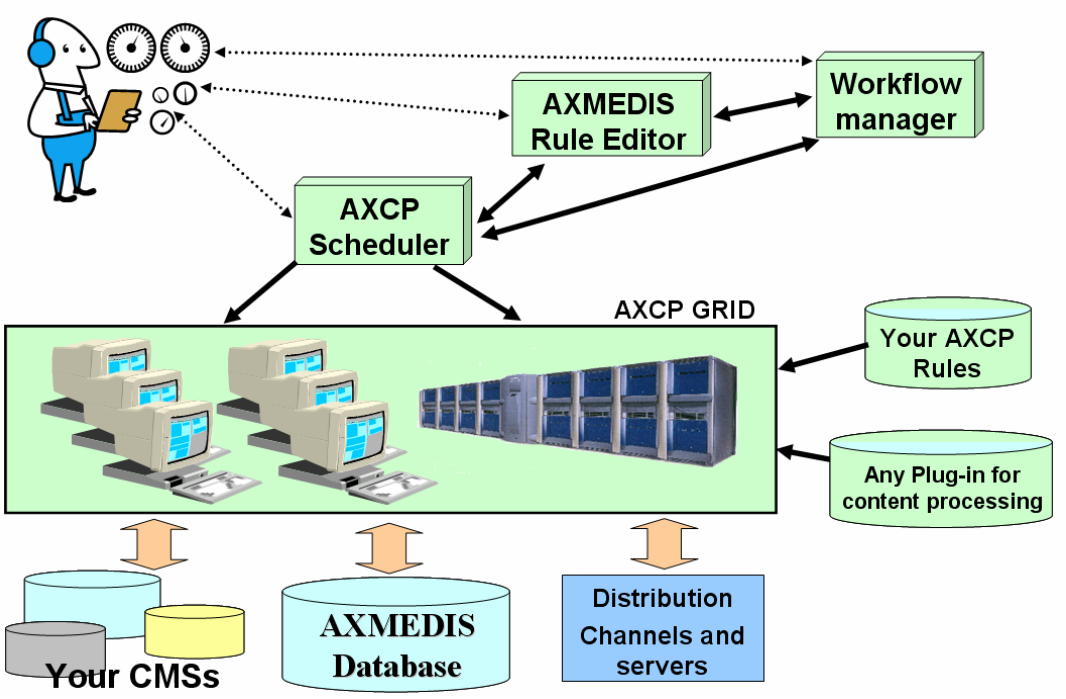
- **Content Ingestion and Gathering** from
  - Content Management Systems, from file system, or protocols;
  - processing resources and coupling them with metadata;
- **Content Retrieval** from
  - AXMEDIS database;
  - other AXMEDIS content Factories by means of the P2P tools, namely AXEPTool;
  - Content Management Systems, CSMs;
- **Content Storage** into
  - an AXMEDIS database;
  - ODBC based databases, other databases via Web Services, and other models will added later;
- **Content Processing** such as
  - digital resources adaptation, extraction of descriptors, transcoding, synchronisation, metadata processing, estimation of fingerprint, watermarking, indexing, content summarization, etc.;
  - metadata manipulation, mapping and adaptation;
- **Content Composition** for
  - creation of content components or objects as a combinations of raw assets such as Text, Images, Audio, Video (actual shot), Animation (synthetic), metadata, descriptors, licenses, multimedia objects such as MPEG-4, HTML, SCORM, macromedia tool file, animations, games, etc.;
  - creation of content as linear or hierarchical combination of content components;
- **Content Formatting**
  - gluing content elements together by means of SMIL based templates and applying style sheets to define the usage interface (format, layout) of the whole collection of content elements and the interested content usage paradigms (leaving open some parameters). For example, karaoke, collection browsing, selection menus, sliding presentation, stable background with a window with live video, animated text moving on an image, running text, etc.;
  - optimization of styling parameters left open or defining them manually to arrange for example: best fitting of images in the screen, optimizing the amount of text in the page, best time fitting, etc.;
- **Content Protection** such as:
  - protection of digital resources and full objects with their complex structure;
  - creation of Protection Information parameters, such as keys, or other features;

  - applying Protection Information model to AXMEDIS object, segmenting digital resources, slicing objects, applying encryption, scrambling, compression, and many other algorithms;
  - posting specific protection information for each given object to the AXMEDIS Certifier and Supervisor server;
- **Content Licensing** for
  - generating license from license model and additional information, storing licenses, posting them on license server automatically;
  - transcoding/translating licenses;
  - invocation of some verification algorithms about licenses and available rights to simulate the usage from the user site;
- **Content Publication and Distribution** towards
  - any distribution channel, producing programme and its schedule;
  - P2P network of other AXMEDIS Factories of content integrators, producers, and distributors.

3.4 AXMEDIS Content Processing Tools

In order to exploit the above list of features to process/manipulate content, resources, licenses, XML, SMIL, databases, protection information, etc., the AXMEDIS Content Processing Area is governed by a set of tools:

- **AXMEDIS Rule Editor**: to produce, debug, test and validate the executable AXCP Rules that can be:
  - written with a simple **AXCP language for content production** which is an extension of Java script;
  - created as macros from AXMEDIS Editor and authoring tools;
  - tested, debugged and validated on the AXMEDIS Rule Editor;
  - activated for content processing on any AXCP GRID Node or on a single computer;
  - used for B2B or B2C purpose;
  - used/parameterized for producing content on demand or to be integrated in your content factory;
  - activated from your Workflow Manager engine via web service;
  - activated by changes in remote objects and queries in the local database and on the P2P network.
- **AXCP GRID**: a set of general purpose or specialized computers to execute AXCP Rules governed by the AXCP Rule Scheduler;
- **AXCP GRID Node**: a single general purpose or specialized computer of the AXCP GRID;
- **AXCP Scheduler**: to schedule the AXCP rule of GRID nodes according to the content production and processing needs in terms of time and resources.



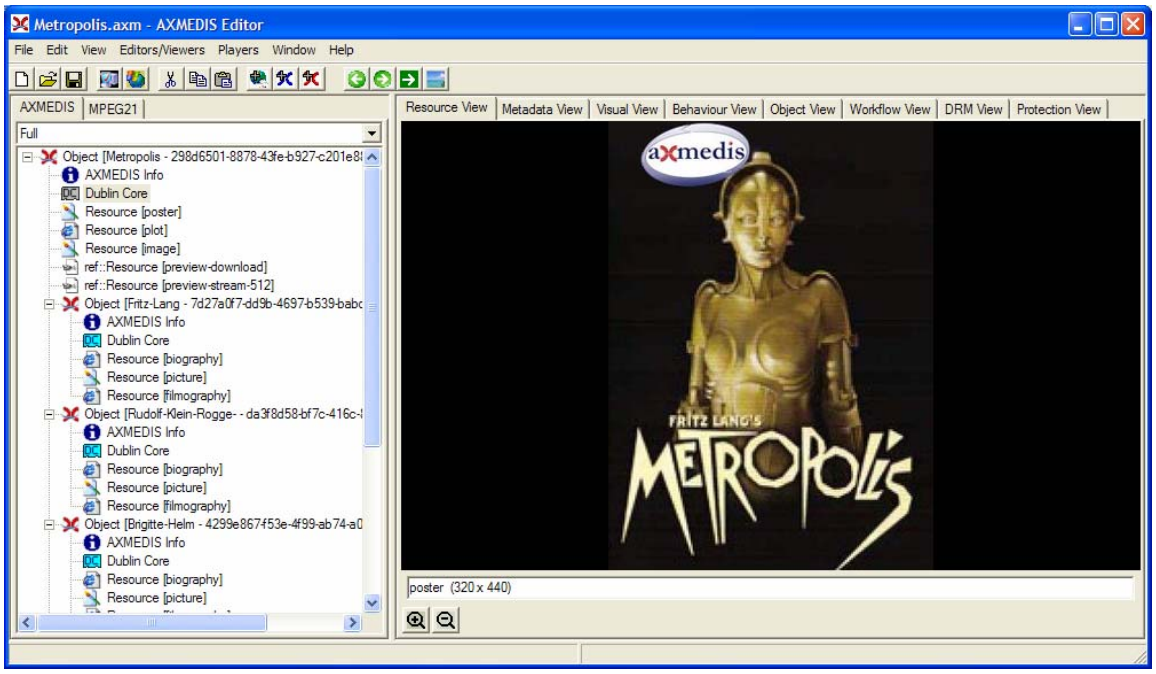
The processing capabilities and functionalities reported in the previous section and that can be exploited from AXCP Tools and Rules can be simply expanded by means of realizing or installing a set of additional plug-ins. The AXMEDIS Plug-in technology is open since:

- the specification of plug-in format is public;
- a plug-in tool kit (including examples and source code for creating those plug in) are public and accessible to all;
- a large part of the above mentioned features are provided by means of AXMEDIS Plug-ins. This demonstrate the solution flexibility;
- any user or third party company can create its own plug-in or include in a plug-in any open third party library including those open source.

3.5 AXMEDIS Editors, the authoring tools

AXMEDIS Editor is used for the manual production of AXMEDIS Objects, and allows creating and manipulating object features and their different aspects:

- ***Structure***, to add, remove, move digital resources and metadata inside the AXMEDIS Object (hierarchy editor and viewer of the object);
- ***Resource manipulation***, to use content processing plug-ins for generic and customizable resource manipulation (the same algorithms and tools used in the AXMEDIS Content Processing);
- ***Metadata editing***, allowing to edit/mapping the metadata related to digital resources and objects (metadata editor and viewer);
- ***Visual and behavioral editing,*** allowing to define content usage paradigm with SMIL organization of resources to present/layouting the digital content contained inside the AXMEDIS Object (visual and behavioral editor and viewer);
- ***Protection editing,*** allowing to specify the protection algorithms to be used for the AXMEDIS Object protection, and thus to define the Protection Information (Protection Editor and Viewer);
- ***DRM editing***, allowing to produce and verify licenses for end users and/or distributors of the AXMEDIS Object as well as the Potentially Available Rights, PAR, that could be acquired on objects shared in the P2P Network (DRM editor and viewer);
- ***Workflow***, allowing editing and viewing the status and the work to be done on the AXMEDIS Objects involved in the workflow process (workflow editor and viewer).



Moreover, **AXMEDIS Editor:**

- allows to perform queries to look for content to produce AXMEDIS Objects integrating other AXMEDIS objects coming from the internal AXMEDIS Database, the AXMEDIS P2P network or possibly from the factory CMS;
- allows download/upload AXMEDIS Objects from/to the AXMEDIS Database and file system;
- can be controlled by the AXMEDIS workflow system to integrate manual operations on AXMEDIS objects inside the production process formalized with the workflow system;
- can be used to inspect automatically generated objects for validation;
- can finalize the production of automatically produced objects;
- can produce rules/script to be used as templates for the AXMEDIS Content Processing environment.

3.6 Content Adaptation facilities

AXMEDIS objects can be created to be distributed over heterogeneous networks and towards different kind of terminals, client tools/devices. Moreover, the people who will ultimately consume and interact with the content may have different behavior and preferences, and the best formats to provide them the best experience on their terminal could be different. Consequently, digital items may need to be adapted to fit any particular usage environment. This is the goal of AXMEDIS content adaptation tools which aim at achieving interoperable transparent access to (distributed) advanced multimedia content by shielding Users from network and terminal installation, management and implementation issues.

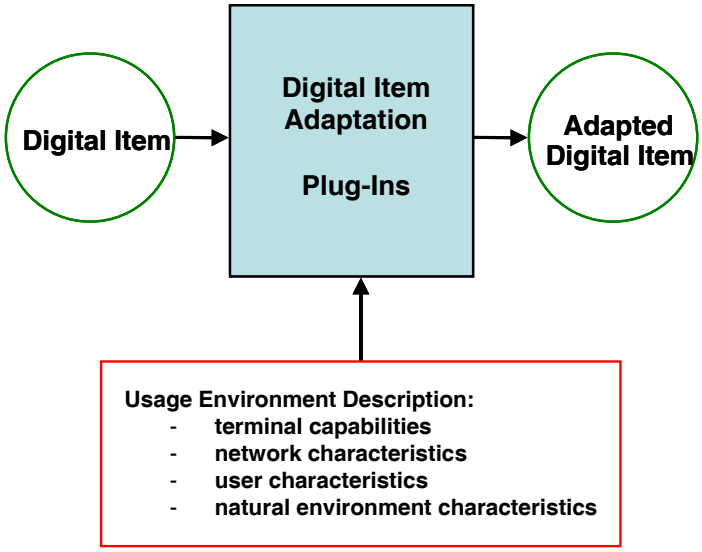
Adaptation may involve:

- transcoding of digital resources that means to change the format (for example, from TIF to GIF, from MPEG-4 to video, from a video to MPEG-4, from MPEG-4 video to a MPEG-2, from audio to symbolic music, from audio to MIDI, from audio in PCM to a ringtone format for mobiles, etc.), re-sampling, shrinking, stretching; In some cases, some features related to resolution, interactivity, are lost in the process in favor of having the content usable in another format. Typically the content is produced in a format that is transcoded scaling down its features and not the vice versa;
- manipulation of licenses, reduction of license scope, reduction of time or territorial information, transformation of format, translation of license (such as passing from ODRL to MPEG-21 REL), etc.;
- manipulation of metadata, metadata mapping, metadata reduction, translation of metadata, etc.

The **adaptation process** can be performed:

- during the content production (for digital resources, licenses and metadata) by exploiting functionalities accessible from the AXMEDIS Editors and/or for the AXMEDIS Content Processing tools
- directly on the player terminal/device (mainly for digital resources and metadata). In this case, ISO/IEC 21000 (MPEG-21) specified a set of normalized tools for the adaptation of digital content describing the usage environment of a digital item to command adaptation tools. Within the AXMEDIS players, MPEG-21 usage environment descriptions are used to drive the adaptation tools considering:
  - Terminal capabilities (codec, formats, input-output, etc., supported by the terminal),
  - Network characteristics (for example, the minimum guaranteed bandwidth of a network),
  - User characteristics (presentation preferences, auditory or visual impairment etc.),
  - Natural environment characteristics (for example, the illumination characteristics that may affect the perceived display of visual information).

The conceptual architecture of the adaptation engine is shown below: a digital item is subject to adaptation thanks to dedicated plug-ins to produce the adapted digital item; the adaptation performed by the plug-ins is parameterized according to MPEG-21 usage environment descriptions.

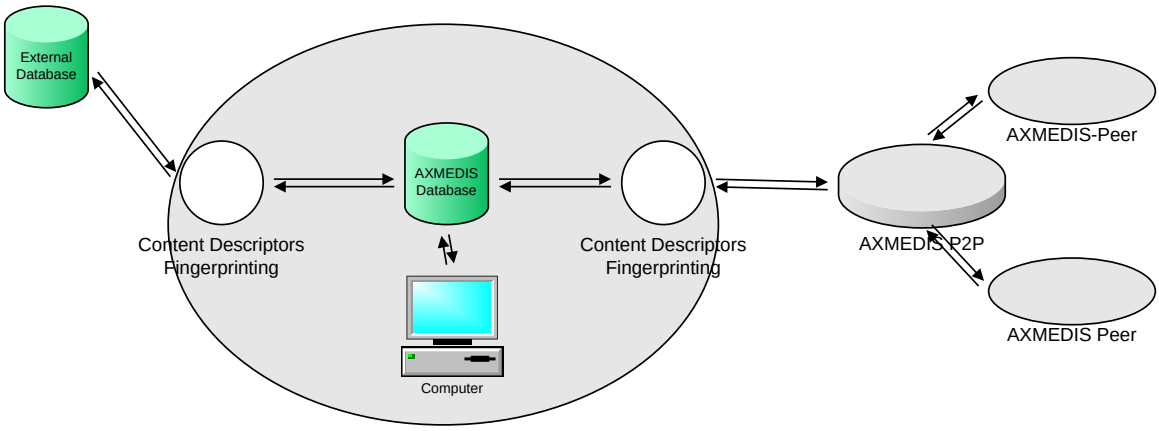


3.7 Content Fingerprint and Descriptors extraction

Among the several content processing algorithms that can be applied in AXMEDIS, a large set can be classified as extractors or estimators of Fingerprint and/or Descriptors of content.

Content fingerprints and descriptors can be used in the AXMEDIS framework for different purposes:

- Classification and recognition of content and/or digital resources;
- Identification of a single piece of content/resource and thus for content certification;
- Authentication/verification of content integrity.



They can be of classified as:

- **High-level descriptors** to describe content with a set of high-level features independent on the format and content resolution. They are typically high level features immediately related with concepts understandable by humans, such as: rhythm and tonality for music, subject for text, etc. They can be very easily used as Descriptors by humans to make queries in the data bases, and thus to retrieve similar content;
- **Digital fingerprints** (or **perceptual hash values**). They can be compared with human fingerprints and may be used to identify a specific content and are robust against data transformation;
- **Low-level descriptors** to describe digital information at lower level and sometime these descriptors are not independent on the format and resolution, such as: energy for music, spectra for images, dynamic, duration, etc. Similar content may share similar low-level descriptors. In some cases, they are used at the basis to estimate high-level descriptors. They can be used for recognition of content as well;
- **Low-level fingerprints** to estimate a value from a specific digital resource, in many cases estimated with algorithms that do not take into account the content type, such as: cryptographic hash value that can be estimated for any digital file. They can be used for content and digital resource verification of consistency and authentication.

In AXMEDIS, Content Fingerprints and Descriptors can be:

- manually selected and estimated via the AXMEDIS Editors;
- automatically estimated and stored in the object metadata or other places by means of an AXMEDIS Content Processing Rule script. For example: during the acquisition/crawling of content from CMS, during content composition and/or formatting, during any content processing also included in the production on demand;
- estimated to verify integrity of content when the content is opened by an AXMEDIS Editor, or by an AXMEDIS Player, or processed by a AXMEDIS P2P tool such as AXEPTool or AXMEDIA, etc., that is any time that an AXMEDIS object is loaded into the AXOM core component of AXMEDIS Framework.

4 AXMEDIS Content Processing Script Language User Manual

4.1 AXOM, AXMEDIS Model and Javascript classess

The set of JavaScript Classes that wraps the main classes of the AXMEDIS Object Model includes the:

AxmedisObject - for managing an instance of the AXMEDIS Object
AxInfo - for managing the metadata of the AXInfo section.
AxDublinCore - for managing an instance of Dublin Core metadata section
AxResource - for managing digital resources embedded or to be embedded in the AXMEDIS bject.
AxMetadata - for managing an instance of an AxMetadata object to cope with generic metadata
Example - Example of usage and scripts

4.1.1 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

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)*
0004	Performs encryption and decryption using Blowfish	0. string *key* 1. string *iv* 2. int16 *keyLen* 3. int16 *ivLen*
0005	Performs encryption and decryption using AES	
0006	Performs encryption and decryption using 3DES	
0007	Performs encryption and decryption using CAST-128	

Please refer to the cryptlib manual for further implementation details. (www.cryptlib.com)

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.

4.1.2 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
number distributorCount
number of Distributors present

// 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
string getOwnerID()
allow getting the code identifying the owner

setOwnerID(string value)
allow setting the code identifying the owner

string getOwnerIDCoding()
allow getting the coding used to identify the owner

setOwnerIDCoding(string value)
allow setting the coding used to identify the owner

string getOwnerName()
allow getting the name of the owner

setOwnerName(string value)
allow setting the name of the owner

string getOwnerURL()
allow getting the URL of the owner

setOwnerURL(string value)
allow setting the URL of the owner

string getOwnerCompany()
allow getting the company of the owner

setOwnerCompany(string value)
allow setting the company of the owner

string getOwnerCompanyURL()
allow getting the company URL of the owner

setOwnerCompanyURL(string value)
allow setting the company URL of the owner

string getOwnerNationality()
allow getting the nationality of the owner

setOwnerNationality(string value)
allow setting the nationality of 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
addDistributor()
adds a Distributor if not present.

removeDistributor()
removes the Distributor

number getDistributorCount()
returns the number of Distributors present

string getDistributorAXDID()
allow getting the AXDID value for the Distributor

setDistributorAXDID(string value)
allow setting the AXDID value for the Distributor

string getDistributorName()
allow setting the Name value for the Distributor

setDistributorName(string value)
allow getting the Name value for the Distributor

string getDistributorURL()
allow getting the URL value for the Distributor

setDistributorURL(string value)
allow setting the URL value for the Distributor

string getDistributorNationality()
allow getting the Nationality value for the Distributor

setDistributorNationality(string value)
allow setting the Nationality value for the Distributor

Object Status
string getAccessMode()
it returns the access mode: “READ_ONLY” or “READ_WRITE” string.

setAccessMode(string value)
allow setting the Access the object specifying strings “READ_ONLY” or “READ_WRITE”

Date getCreationDate()
get the local date and time of object creation

Date getLastModificationDate()
get the local date and time of object modification

number getVersion()
get the version of the object

number getRevision()
get the revision of the object

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.

string getObjectType()
allow to get object type (“BASIC” or “COMPOSITE”)

boolean getObjectIsProtected()
allows getting if the object is protected or not. True means protected.

setObjectIsProtected(Boolean value)
allows to get and set if the object is protected or not

string getProtectionStamp()
allows getting the protection stamp

setProtectionStamp(string value)
allows setting the protection stamp

boolean getObjectIsGoverned()
allow getting 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

setObjectIsGoverned(in value:bool)
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

Workflow Status
string getWorkflowWorkItemID()
allow getting the WorkflowWorkItemID

setWorkflowWorkItemID (string value)
allow setting the WorkflowWorkItemID

string getWorkflowWorkspaceInstanceID()
allow getting the WorkflowWorkspaceInstanceID

setWorkflowWorkspaceInstanceID (string value)
allow setting the WorkflowWorkspaceInstanceID

Internal Potential Available Rights Management
addInternalPotentialAvailableRights()
adds a new Internal PAR section

removeInternalPotentialAvailableRights()
removes the Internal PAR section

getInternalPotentialAvailableRightsCount()
gets how many Internal PAR sections are present (0 or 1)

string getInternalPotentialAvailableRightsStatus()
allow getting the internal PAR status

setInternalPotentialAvailableRightsStatus(string value)
allow setting the internal PAR status

License getInternalPotentialAvailableRightsLicense()
gets the license

Potential Available Rights Management
addPotentialAvailableRights()
adds a new PAR section if not present

removePotentialAvailableRights()
removes the PAR section

getPotentialAvailableRightsCount()
gets how many PAR sections are present (0 or 1)

string getPotentialAvailableRightsLicensingURL()
allow getting the licensing URL

setPotentialAvailableRightsLicensingURL (string value)
allow setting the licensing URL

string getPotentialAvailableRightsStatus()
allow getting the PAR status

setPotentialAvailableRightsStatus(string value)
allow setting the PAR status

License getPotentialAvailableRightsLicense()
gets the license

Object History Management
string getHistoryOfVersion(number version)
gets the history of a version as an XML string

4.1.3 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

load (string path)
It loads a Raw Resource from filesystem

save (string path)
It saves a Raw Resource onto filesystem

4.1.4 AxDublinCore

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

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.

4.1.5 AxMetadata

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
• store the XML of metatada

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

string getMetadataID ()
It returns the string with the ID of metadata

setMetadataID (string id)
It sets the ID of metadata

4.1.6 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();
```

4.2 Network

Javascript connection classes are a set of 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:

Form - The class models a simple form of a http web page
HttpConnection - The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
FtpConnection - The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
OdbcConnection - The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
WebServiceConnection - The class models a meta class for managing web services by loading WSDL description and dynamically creating services (methods).
Example - Example of usage and scripts

4.2.1 Form

This class models a 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

4.2.2 HttpConnection

The class provides the support to the http connection.

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

boolean 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.

boolean clearLogin()
Reset the account parameters for login

boolean setPort(string port)
set a Port at the port number value

boolean clearPort()
Clear the port number resuming the default port.

boolean getToString(string url)
Get function, copy content at the specified url to a String. It is used in conjuction with getContent().

boolean getToFile(string url, string filePath)
Get function, copy content at the specified url to a File at filePath

boolean putFromFile(string url, string filePath)
Put function, put a content From a File at filePath to a specified url

boolean putFromString (string url, string buffer)
Put function, put content From a String buffer to a specified url

boolean 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.

4.2.3 FtpConnection

The class provides the support to the Ftp connection.

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.

4.2.4 OdbcConnection

The class provides the support to an Odbc based connection

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.

4.2.5 WebServiceConnection

Class name in Javascript is WebServiceConnection

Exposed Propertie

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.

4.2.6 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”);
```

4.3 SearchBox Javascript Classes

The searchbox bridge provides the following set of JavaScript 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
Example - Example of usage and scripts

4.3.1 AXSearchbox

The main object for accessing searchbox.

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.

4.3.2 Document
Used to handle documents as opaque objects.

Exposed Properties:
mimeType [string]
MIME type of the document.

size [integer]
size (in bytes) of the document.

Exposed 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.

4.3.3 MetadataValue
Used to hold metadata.

Exposed Properties:
key [string]
Metadata key.

slice [integer]
Slice where metadata is stored.

value [string]
Metadata value.

4.3.4 DocumentMetadata
Used to describe the metadata associated with a document.

Exposed 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.

4.3.5 QueryParser
Used to specify the query string parser to use.

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.

4.3.6 QueryInfo
Used to specify the information returned as result of a query.

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.

4.3.7 QueryView
Used to restrict the set of documents returned as result of a query.

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.

4.3.8 QuerySort
Used to specify the sorting of documents returned as result of a query.

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.

4.3.9 QueryAtomType

Used to specify the type of a QueryAtom.

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.

4.3.10 QueryAtom
Used to build a query in RPN notation.

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.

4.3.11 QuerySliceWeight
Used to specify slice weights.

Exposed Properties:

slice [integer]
Dict ID. The following dict IDs can be used:
1. Author
2. Keyword
3. Abstract
4. Invisible
5. Marginal normal text
6. Marginal emphasized text
7. Marginal link text
8. Marginal remote link text
8. Marginal header text
10. Central normal text
11. Central emphasized text
12. Central link text
13. Central remote link text
14. Central header text

15. Title

weight [integer]
Slice weight. Must be greater or equal to 0.

4.3.12 QuerySpec

Used to submit a query.

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.

4.3.13 QueryResult

Used to return information on a query result.

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.

4.3.14 Examples of usage

```
var sb = new AXSearchbox();
sb.host = "localhost";
sb.port = "2200";
sb.username = "admin";
sb.password = "password";
var qs = new QuerySpec();
var a = new Array(1);
a[0] = 1;
qs.archives = a;
qs.parser = QueryParser.ALGPARSER;
qs.info = QueryInfo.INFO_CONTEXT;
qs.view = QueryView.VIEW_PUBLISHED;
qs.sort = QuerySort.SORT_STANDARD;
qs.queryString = "string";
```

```
qs.firstDoc = 0;
qs.lastDoc = 1;
var qr = new Array();
var maxres = sb.query(qs, qr);
var i, j; for(i = 0; i < qr.length; ++i)
{
   print(qr[i].id+" "+qr[i].url);
   var doc = sb.getDocument(qr[i].id);
   var links = sb.getDocumentOutlinks(qr[i].id);
   var meta = sb.getDocumentMetadata(qr[i].id);
   var metavalues = meta.getAllValues();
   print("-> "+doc.mimeType+" ["+doc.size+"]");
   for(j = 0; j < links.length; ++j)
   {
      print("O-> "+links[j]);
   }
   if(metavalues != null)
   {
      for(j = 0; j < metavalues.length; ++j)
      {
         print("M-
         >"+metavalues[j].key+"["+metavalues[j].slice+"]="+metavalues[j
         ].value);
      }
   }
}
```

4.4 License and Javascript Classes

JavaScript License classes are a set of classes that provide the support of the license and PAR to the AXCP Engine by modelling in terms of primitive functionalities the license and the PAR classes. They consist in:

Condition - This class is the abstract class for all conditions related to license.
Fee - Fee is a type of condition.
Grant - A Grant is a container element that relates to a principal with a right over a resource. It can contain some conditions.
Principal - The Principal is the entity that will be able to exercise the right.
GrantGroup - A GrantGroup is a virtual container of Grants.
Interval - An Interval is a type of condition. It express valid interval of time to exercise the right.
Issuer - Issuer is the creator of the MPEG21 REL licence. It is the entity that provides the rights.
License - License is the certificate provided to the user for particular content.
LicenseManager - LicenseManager is the class that manages the license database.
LicensePattern - This class manages the pattern for the license.
OMALicense - The license object contains all the information related to an OMA license (such as permissions, rights, etc.).
Number - Number is a type of condition of a MPEG21 REL license. It expresses how many times the right can be exercised.
Resource - Resource is the object over which the right can be exercised on.
Right - Right expresses the action that can be taken to an object.
Territory - Territory is a type of condition.
PAR - This class provides the model for a PAR object.
PARGrant - A PARGrant is a container element that relates to a principal with a right over a resource. It can contain some conditions.
PARGrantGroup - A PARGrantGroup is a virtual container of grants. It is the way that MPEG21-REL uses to define a set of grants.
PARResource - Resource is the object on which the right can be exercised. It is a part of the grant of an AXMEDIS PAR. The resource expresses an object. Depending of the right, this object can be different. If the right is "issue", the Resource is a GrantGroup. If the right is not "issue", the Resource contains the identifier.
PARRight - Right is a part of the grant of a MPEG21 REL license. There are a set of rights that has been

declared by MPEG21 REL.
PMSClient - A PMSClient is the client interface for the PMS. It is responsible for distributing all the protection related jobs amongst the different modules and the PMS server.
Contract - Represents a human-readable contract generated from a license. In practice, classes derived from Contract will be used, whereas this class is an abstract class (pure virtual).
RightsExpressionTranslator - This class provides method for converting license formats.

In order to avoid too many parameters for the wrapper methods the following classes were created, which provide JS Objects that can be used as parameters in JavaScript.
The actual parameters are extracted from these objects in the wrapper method and passed to the C++ class methods.

RightsParameters - This class covers the parameters that are related to the rights granted to a user by a license.
RightsFeeParameters - This class covers the parameters that are related to the fee affiliated with rights.
RightsIntervalParameters - This class covers the parameters that are related to the time interval mentioned in the rights granted to user.
RightsLimitParameters - This class covers the parameters that are related to the number of times that each right can be used.
RightsRegionParameters - This class covers the parameters that are related to specific region specified in the rights.
PARGrantParameters - This class covers the parameters that are related to PARGrant.
AxToolInfoParameters - This class covers the parameters that are related to AXMEDIS Tool.
ProtInfoParameters - This class covers the parameters that are related to AXMEDIS Protection Information.
AXIDParameters - This class covers the parameters that are related to different AXMEDIS identifiers (IDs).
OpInfoParameters - This covers the parameters that are related to operation and execution information.

4.4.1 Condition
This class is the abstract class for all conditions related to license.

Exposed Properties

tipo
Stores the type of condition.

Exposed methods

TypeCondition getTypeCondition ()
This method returns the condition fee, territory, number or interval.

4.4.2 Fee
Fee is a type of condition.

Exposed methods

char getServiceReference ()
Returns the service reference value.

*void setServiceReference(char *c)*
Returns the service reference value.

*char *getCurrency(void)*
This method returns the type of the currency.

*bool setCurrency(char *c)*

This method establishes the type of the currency.

float getAmount(void)
This method returns the amount of the fee.

void setAmount(float c)
This method establishes the amount of the fee.

*char *getTo(void)*
This method returns "TO MPEG21" REL element.

*void setTo(char *c)*
This method establishes "TO MPEG21" REL element.

FeeType getFeeType()
This method returns the type of fee.

void setFeeType(FeeType f)
This method establishes the type of fee.

4.4.3 Grant

A Grant is a container element that relates to a principal with a right over a resource. It can contain some conditions.

Exposed methods

*void addCondition(Condition *c)*
This method adds a condtion to the vector with conditions of the grant.

Principal getPrincipal(void)
This method returns the principal.

void setPrincipal(Principal p)
This method establishes the principal of the grant.

Right getRight(void)
This method returns a copy of the right of the Grant.

void setRight(Right p)
This method establishes the right of the license.

Resource getResource(void)
This method returns the resource of the grant.

void setResource(Resource p)
This method establishes the resource of the grant.

vector getConditions()
This method returns a copy of the vector of conditions of the grant.

string getId(void)
This method returns the ID.

void setId (string i)
This method establishes the ID.

string getLicId(void)
This method returns the license Temperal Id.

void setLicId(string i)

This method establishes the license Temperal Id.

4.4.4 Principal

The Principal is the entity that will be able to exercise the right.

Exposed methods

char getName(void)*
This method returns the principal identifier or name.

*void setName(const char *n)*
The method establishes the principal identifiers.

4.4.5 GrantGroup

A GrantGroup is a virtual container of Grants.

Exposed Properties

Exposed methods

vector& getGrants()
This method returns the vector of the reference to the grants (it is a constant reference).

void addGrant(Grant g)
This method adds a new grant to the vector of grants.

string getId(void)
This method returns the id.

void setId(string i)
This method establishes the id.

4.4.6 Interval

An Interval is a type of condition. It expresses a valid interval of time to exercise the right.

Exposed Properties

notBefore_prop notAfter_prop

Exposed methods

setNotBefore(string date)
This method establishes the beginning date of the interval.

string getNotBefore()
This method returns beginning date of the interval.

setNotAfter(string enddate)
This method establishes the ending date of the interval.

string getNotAfter() This method returns the ending date of the interval.

4.4.7 Issuer

Issuer is the creator of the MPEG21 REL licence. It is the entity that provides the rights.

Exposed Properties

issue_time,details,issuer,id

Exposed methods

*char *getIssueTime(void)*
This method returns the time of issuance (when the license was created).

*void setIssueTime(char *c)*
This method set the time of issuance (when the license was created). Format : YYYY-MM-DDTHH:MMSS

*char *getIssuer(void)*
This method returns the issuer.

*void setIssuer(char *c)*
This method establishes the issuer.

*char *getDetails(void)*
This method returns some details, additional information.

*void setDetails(char *c)*
This method establishes the details of the issuance.

string getId(void)
This method returns the ID of the issuer.

void setId(string i)
This method establishes the ID of the issuer.

4.4.8 License

License is the certificate provided to the user for particular content.

Exposed Properties

Exposed methods

void addGrant(Grant g)
This method adds a grant to a grant group.

Issuer getIssuer(void)
This method returns the issuer of the license.

void setIssuer(Issuer i)
This method establishes the issuer of the license.

Grantgroup getGrantgroup()
This method returns a copy of the grant group of the license.

void setGrantGroup(GrantGroup gg)
This method establishes the grant group of the issueance.

string 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.

License RetrieveLicenseFromFile(string filepath)
This method retrieves the license from a file.

StoreLicenseToFile(string filepath)
This method stores the license in the file.

4.4.9 LicenseManager

LicenseManager is the class that manages The license database.

Exposed Properties

Exposed methods

string StoreLicense(License lic)
This method stores the license.

string StoreLicense(string lic)
This method stores the license. It is the overloaded method.

string StoreLicense(License lic, string licenseDBid)
This method stores the license. it is the overloaded method which takes the two arguments mentioned above.

string 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.

License RetrieveLicenseFromFile(string filepath,int &errorcode)
This method retrieves the license from a file.

License StringXMLLicense2LicenseModel(string XMLLicense,int &errorcode)
This method converts the license from a string to a license model.

PAR RetrievePARFromFile(string filepath,int &errorcode)
This method retrieves the PAR from a file.

PAR StringXMLPAR2PARModel(string XMLLicense,int &errorcode)
This method converts the license from a string to a licensemodel.

string StorePAR(PAR parp)
This method stores the PAR.

string StorePAR(string parp)
This method stores the PAR. It is an overloaded method.

string RetrieveLicense(string licenseId)
This method retrieves the license.

string RetrievePAR(string PARId)
This method retrieves the PAR.

*int DataLicense(string licenseId,char **p,int *s);*
This method gets data from license.

string LicenseManager StorePAR(string AXOID,string par)
This method stores the PAR.

string LicenseManager StorePAR(string AXOID,PAR parp)
This method stores the PAR.

string LicenseManager::RemovePAR(string AXOID)
This method removes the PAR.

4.4.10 LicensePattern

This class manages the pattern for the license.

Exposed Properties

Exposed methods

License generateLicenseFromPattern(string AXUID,string AXOID)
This method generates a new license object from a license pattern changing the principal of the grants to AXUID and the resource to AXOID.

4.4.11 OMALicense

The OMAlicense object will contain all information related to an OMA license (such as permissions, rights, etc.).

Exposed Properties

Exposed methods

4.4.12 Number

Number is a type of condition of a MPEG21 REL license. It expresses how many times the right can be exercised.

Exposed Properties

serviceReference count

Exposed methods

SetServiceReference(string sreference)
This method establishes the service reference.

string sreference getServiceReference()
This method returns the service reference.

setCount(int count)
This method establishes number of counts that the right can be exercised.

int getCount()
This method returns the number of times that the right can be exercised.

4.4.13 Resource

Resource is the object on which the right can be exercised.

Exposed Properties

resourceID,item,ggResource,tipo,diType

Exposed methods

string getResourceID()
This method returns the resource (digital item) identifier.

string setResourceID(setResource id)
This method establishes resource (digital item) identifier.

string getItem()
This method returns the item identifier (if existing).

setItem(Strimg item)

This method establishes the item identifier. It is useful to select one item of the whole digital item.

Grantgroup getGrantGroup()
This method returns the grant group of the identifier.

setGrantGroup(GrantGroup ggroup)
This method establishes the grant group if the right is "issue".

isNullGrantGroup()
This method the grant group if the right is "issue".

GetTipo()
This method returns the resource type (DI or grant group).

4.4.14 Right

Right expresses the action that can be taken to an object.

Exposed Properties

right

Exposed methods

Right getRight()
This method returns the right of the license.

string setRight(setResource id)
This method establishes the right of the license.

4.4.15 Territory

Territory is a type of condition.

Exposed Properties

territory

Exposed methods

string getLocation()
This method returns the location.
Expressed like: "country{Spain}region{Catalonia}..."
Values: country, region, state, city, street, postalCode

string setRight(setResource id)
This method establishes the right of the license.

4.4.16 PAR

This class provides the model for PAR object

Exposed Properties

Exposed methods

addPARGrant(PARGrant g)
This method adds a grant group to the PAR.

setPARGrantGroup(PARGrantGroup gg)
This method establishes the grant group of the PAR.

4.4.17 PARGrant

PARGrant is a virtual container of grants. It is how MPEG21 REL defines a set of grants.

Exposed Properties

Exposed methods

*void addCondition(Condition *c)*
This method adds a condition to the vector of conditions of the grant.

void setRight(Right p)
This method establishes the right of the grant.

void getRight(Right p)
This method returns the right of the grant.

void setPARResource(PARResource p)
This method establishes the resource of the grant.

PARResource getPARResource(void)
This method returns a copy of the resource of the Grant.

vector getConditions()
Returns a copy of the vector of conditions of the Grant.

void setId(string i)
This method establishes the ID.

string getId(void)
This method returns the ID.

string getId(void)
This method returns the ID.

void setPARId(string i)
This method establishes the PAR temporary ID.

string getPARId(void)
This method returns the PAR temporary ID.

4.4.18 PARGrantGroup

The grant is the basic element to represent the rights offered in an AXMEDIS PAR. The grant relates a principal (owner of the right), a right and a resource, with a set of conditions.

Exposed Properties

Exposed methods

vector getPARGrants()
Returns a reference to the vector of grants. It is a constant reference.

void setId(string i)
This method establishes the ID.

string getId(void)
This method returns the ID.

void addPARGrant(PARGrant g)

This method adds a new grant to the vector of grants of the grant group.

4.4.19 PARResource

Resource is the object on which the right can be exercised. It is a part of the grant of an AXMEDIS PAR. The resource expresses an object.

Exposed Properties

Exposed methods

*char *getPARResourceID(void)*
This method returns the resource (digital item) identifier.

*void setPARResourceID(char *r)*
This method establishes the resource (digital item) identifier.

*char *getItem(void)*
This method returns the item identifier (if existing).

*void setItem(char *i)*
This method establishes the item identifier.

PARGrantGroup getPARGrantGroup(void)
This method returns a copy of the grant group if the right is "issue".

void setPARGrantGroup(PARGrantGroup gg)
This method establishes the grant group if the right is "issue".

TypePARResource getTipo(void)
This method returns the resource type (DI or GRANTGROUP).

4.4.20 PARRight

Right is a part of the grant of a MPEG21 REL license. There are a set of rights that can be declared by MPEG21 REL.

Exposed Properties

Exposed methods

*char *getRight(void)*
This method returns the right.

*bool setRight(char *n)*
This method establishes the right.

4.4.21 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 Properties

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.

string 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. The actual input parameters of the method are
string licenseTmpId
string principal
string AXOID
init diType
init diSubType
string right
bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region
int feeType
float fee
string currency
string bankAccount
string adaptationRules
but to avoid to have too many arguments from the JavaScript, some parameter classes are provided whose summary is given below (at the end).

string 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.

string 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.

string 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. The actual input parameters of the method are
string licenseTmpId
string principal
string AXOID
init diType
init diSubType
string right

bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region
int feeType
float fee
string currency
string bankAccount
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given below.

string 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. The actual input parameters of the method are
string licenseTmpId
string distGrantId
string right
bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region
int feeType
float fee
string currency
string bankAccount
string adaptationRules
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given below (at the end).

string 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.

string 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.

string 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. The actual input parameters of the method are
string PARTmpId
string AXOID
init diType
init diSubType
string right
bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region
int feeType
float fee
string currency
string bankAccount
string adaptationRules
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given below (at the end).

string 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.

string 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.

string 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. The actual input parameters of the method are
string PARTmpId
string AXOID
init diType
init diSubType
string right
bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region

int feeType
float fee
string currency
string bankAccount
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given in the end.

string 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. The actual input parameters of the method are
string PARTmpId
string distGrantId
string right
bool validityInterval
string notBefore
string notAfter
bool countLimit
int limit
bool validityRegion
string country
string region
int feeType
float fee
string currency
string bankAccount
string adaptationRules
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given below (at the end).

string 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.

string getLicense(string licenseId)
The method getLicense retrieve a license from the database (locally or remotely).

string 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.

std::string getPAR(string PARId)
This method retrieve a PAR from the database (locally or remotely).

string 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.

int authorise(AXIDParameters axidpar,OpInfoParameters opinfopar,ProtInfoParameters protinfopar) (
This method verifies and authorise that an action can be exercised. The actual input parameters of the method are
string axcid

string axdid
string axoid
string axtid
string axuid
string axwid
string estimatedHwFingerprint
string executionTimestamp
string objectVersion
string operationDetailsID
string operationID
string ownerName
string protecttionStamp
string protectionInfoRequired
but to avoid two many arguments from the JavaScript, some parameter classes are provided whose summary is given below (at the end).

CertificationResult 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. The actual input parameters of the method are
axid
axrtid
axdom
toolFingerprint
registeredToolFingerprint
regDeadline

VerificationResult 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. The actual input parameters of the method are
axid
axtid
toolFingerprintDigest
but to avoid many parameters a parameters class AXTOOLInfoParameters is provided. its summary is given below (at the end)..

VerificationResult 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. The actual input parameters of the method are
axid
axtid
toolFingerprintDigest
but to avoid many parameters a parameters class AXTOOLInfoParameters is provided. Its definition is given in the end of this document.

VerificationResult 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.

string 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.

bool 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.

int 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. The actual input parameters of the method are
id
version
protstamp
protinfo
but to avoid many parameters from JavaScript, the ProtInfoParameters class is provided in JavaScript. Actual parameters are extracted from this class object.

4.4.22 Contract

Represents a human-readable contract generated from a license. In practice, classes derived from Contract will be used, whereas this class is an abstract class (pure virtual).

Exposed Properties

Exposed methods

bool generateTxtContract(string templatefilename, string contractfilename)
Generates a text contract from a license and a template contract.

4.4.23 RightseExpressionTranslator

This class provides access method for converting license formats.

Exposed Properties

Exposed methods

string generateTranslation(string _license, string _originalRel, string _destinationRel)
This method converts teh "original license" from its REL to the destination one.

*bool setRight(char *n)*
This method establishes the right.

4.4.24 RightsParameters

This class covers the parameters that are related to the rights granted to a user by a license.

Exposed Properties

string licenseTmpId
This property represents the Temparary Licence Identifier.

string AXUIDPrincipal

This property represents the AXUID of the user (user of the license).

string AXOID
This property represents the resource Identifier.

int diType
This property represents type of resource Identifier.

int diSubType
This property represents subtype of resource Identifier.

string right
This property represents The right that will be granted in the license. Can take the following values:
adapt, delete, diminish, embed, enhance, enlarge, execute, install, modify, move, play, print, reduce, uninstall
that correspond to rights described in “MPEG-21 multimedia extension”.

string adaptationRules
This property corresponds to the different condition adaptation rules of the content.

string distGrantId
This property represents the temporary grant identifier.

Exposed methods

string getLicenseTmpId(void) const
This method returns the temporary license Identifier property.

void setLicenseTmpId(const std::string c)
This method establishes the temporary license Identifier property.

string getAXUIDPrincipal(void)
This method returns the AXUIDPrincipal property.

void setAXUIDPrincipal(string axuidprincipal);
This method establishes the AXUIDPrincipal property.

string getAXOID(void)
This method returns AXOID property.

void setAXOID(string axoid);
This method establishes the AXOID property.

int getDiType(void)
This method returns the DiType property.

void setDiType(int d);
This method establishes the Ditype property.

int getDiSubType(void)
This method returns the Disubtype property.

void setDiSubType(int s)
This method establishes the Disubtype property.

string getRight(void)

This method returns the right property

void setRight(string r)

This method establishes the right property.

string getAdaptationRules(void)
This method returns the adaptation rules property.

void setAdaptationRules(string a)
This method establishes the adaptation rules property.

4.4.25 RightsFeeParameters

This class covers the parameters that are related to the Fee affiliated with rights.

Exposed Properties

int feetype
This property shows if a fee has to be paid to exercise the right.

float fee
This property represents the fee to exercise the right..

string currency
This property represents the currency of the fee.

string bankaccount
If the fee type is not 0, this property represents the bank account where the payment will be done.

Exposed methods

int getFeeType(void)
This method returns the fee type property.

void setFeeType(int ftype)
This method establishes the fee type property.

float getFee(void)
This method returns the fee property.

void setFee(float f)
This method establishes the fee property.

string getCurrency(void)
This method returns the currency property.

void setCurrency(string c)
This method establishes the property.

string getBankAccount(void)
This method returns the bank account property.

void setBankAccount(int ftype)
This method establishes the bank account property.

4.4.26 RightsIntervalParameters

This class covers the parameters that are related to the time interval mentioned in the rights granted to user.

Exposed Properties

bool validityinterval

If this property is TRUE the right can be exercised within a time period. If it is FALSE it could be exercised always.

string notbefore
If validityInterval is TRUE, this property corresponds to the date from which the right will be effective. It has to have the next format: YYYY-MM-DDTHH:MM:SS

string notafter
If validityInterval is TRUE, this parameter corresponds to the date until the right will be effective. It has to have the next format: YYYY-MM-DDTHH:MM:SS

Exposed methods

bool getValidityInterval(void)
This method returns the validityinterval property.

void setValidityInterval(bool vinterval)
This method establishes the validityinterval property.

float getNotBefore(void)
This method returns notbefore property.

void setNotBefore(string nbefore)
This method establishes the notbefore property.

string NotAfter(void)
This method returns notafter property.

void setCurrency(string c)
This method establishes the notafter property.

4.4.27 RightsLimitParameters

This class covers the parameters that are related to the no of counts that each right can be use for.

Exposed Properties

bool countlimit
This property shows if the right will be effective for a specific number of uses (TRUE) or could be exercised any number of times.

int limit
If countlimit is TRUE, this property corresponds to the number of times that the right can be exercised.

Exposed methods

bool getCountLimit(void)
This method returns the countlimit property.

void setCountLimit(bool climit)
This method establishes the countlimit property.

int getLimit(void)
This method returns limit property.

void setNotBefore(int l)
This method establishes limit property.

4.4.28 RightsRegionParameters

This class covers the parameters that are related to specific region specified in the rights.

Exposed Properties

bool validityregion
This property shows if the right can be exercised only in a specific region or everywhere.

string country
If validityregion is TRUE, this property corresponds to the country where the right can be exercised.

string region
If validityregion is TRUE, this parameter corresponds to the region where the right can be exercised.

Exposed methods

bool getvalidityregion(void)
This method returns the validityregion property.

void setvalidityregion(bool vregion)
This method establishes the validityregion property.

string getCountry(void)
This method returns country property.

void setCountry(string c)
This method establishes the country property.

string getRegion(void)
This method returns region property.

void setRegion(string r)
This method establishes the region property.

4.4.29 PARGrantParameters

This class covers the parameters that are related to PAR grant.

Exposed Properties

string PARtmpId
This property represents the temporary PAR identifier.

Exposed methods

string getPARTmpId(void)
This method returns the PARtmpId property.

void setPARTmpID(string partmpid)
This method establishes the the PARtmpId property.

4.4.30 AXToolInfoParameters

This class covers the parameters that are related to AXMEDIS Tool ID.

Exposed Properties

string axtid
This property represents the identifier of the registered AXMEDIS tool ID.

string axrtid
This property corressponds to the registered AXMEDIS Tool.

string toolfingerprint

This property corresponds to the full fingerprint (software and hardware parts) of the installed tool

string regdeadline
This property corresponds to the registration deadline of the installed tool.

byteArray toolfingerprintdigest
This property corresponds to the md5 hash of the full fingerprint (software and hardware parts) of the installed tool.

Exposed methods

getAxtid(void)
This method returns the axtid property.

void setAxtid(string ax)
This method establishes the axtid property.

string getAxrtid(void)
This method returns axrtid property.

void setAxrtid(string ax)
This method establishes the axrtid property.

string getToolFingerPrint(void)
This method returns the toolfingerprint property.

void setToolFingerPrint(string r)
This method establishes the toolfingerprint property.

string getRegDeadLine(void))
This method returns the regdeadline property.

void setRegDeadLine(string rdeadline)
This method establishes the regdeadline property.

byteArray getToolFingerPrintDigest(void)*
This method returns the toolfingerprintdigest property.

void setToolFingerPrintDigest(byteArray tfingerprintdigest)
This method establishes the toolfingerprintdigest property.

4.4.31 ProtInfoParameters

This class covers the parameters that are related to Protection Information

Exposed Properties

string id
This property corressponds to the AXMEDIS object ID.

string version
This property corressponds to the AXMEDIS Obejct version.

string protstamp
This property corresponds to the protection stamp

string protinfo
This property corresponds to the protection information to be updated.

bool protinforeq

This property shows if the object is protected.

Exposed methods

string getId(void)
This method returns the axtid property.

void setId(string ob)
This method establishes the axtid property.

string getVersion(void)
This method returns version property.

void setVersion(string ax)
This method establishes the version property.

string getProtStamp(void)
This method returns the protstamp property.

void setProtStamp(string pstamp)
This method establishes the protstamp property.

string getInfo(void))
This method returns the protinfo property.

void setProtInfo(string pinfo)
This method establishes the protInfo property.

bool getProtInfoReq(void)
This method returns the protinfreq property.

void setProInfoReq(bool pinforeq)
This method establishes the protinforeq property.

4.4.32 OpInfoParameters

This class covers the parameters that are related to Protection Information

Exposed Properties

string estimatedHwFingerprint
This property corressponds to the estimated hardware Finger Print.

string executionTimeStamp
This property corressponds to the execution time stamp.

string operationDetailsID
This property corresponds to the operations details ID.

string ownerName /em>
This property corresponds to the owner name.

string operationID
This property represents the operation ID.

Exposed methods

string getEstimatedHwFingerPrint(void)
This method returns the estimatedHwFingerprint property.

void setEstimatedHwFingerPrint(string ehwfprint)
This method establishes estimatedHwFingerprint property.

string getExecutionTimeStamp(void)
This method returns executionTimeStamp property.

string getExecutionTimeStamp(void)
This method establishes the version property.

void setExecutionTimeStamp(string extimestamp)
This method returns the executionTimeStamp property.

string getOperationDetailsID(void)
This method returns the operationDetailsID property.

void setOperationDetailsID(string opdetailsid)
This method establishes the operationDetailsID property.

string getOperationID(void) const
This method returns the operationID property.

void setOperationID(string opid)
This method establishes the operationID property.

void string getOwnerName(void)
This method returns the ownerName property.

void setOwnerName(string oname)
This method establishes the ownerName property.

4.4.33 AXIDParameters

This class covers the parameters that are related to different AXMEDIS IDs

Exposed Properties

string axcid
This property corressponds to the creator ID.

string axdid
This property corressponds to the distributor ID.

string axoid
This property corresponds to the AXMEDIS object ID.

string axtid /em>
This property corresponds to the AXMEDIS Tool ID.

string axuid
This property represents the AXMEDIS resource ID.

string axwid
This property represents the work ID.

Exposed methods

string getEstimatedHwFingerPrint(void)
This method returns the estimatedHwFingerprint property.

void setAXCID(string ehwfprint)

This method establishes axcid property.

string getAXCID(void)
This method returns axcid property.

string getAXDID(void)
This method establishes axdid property.

void setAXDID(string extimestamp)
This method returns the axdid property.

string getAXOID(void)
This method returns the axoid property.

void setAXOID(string opdetailsid)
This method establishes the axoid property.

string getAXTID(void) const
This method returns the axtid property.

void setAXTID(string opid)
This method establishes the axtid property.

void string getAXUID(void)
This method returns the axuidproperty.

void setAXUID(string oname)
This method establishes the axuid property.

void string getAXWID(void)
This method returns the axwid property.

void setAXWID(string oname)
This method establishes the axwid property.

4.5 Selection Javascript Classes

Brief description of classes

List of classes and link to internal sections

4.5.1 Class Name

Description

Exposed properties:

<type> <attribute name>
description

<type> <attribute name>
description

<type> <attribute name>
description

Exposed methods:

metod(...)
Method with no return. Description of paramers

4.5.2 Examples of usage

```
Put here source code in Javascript to show the usage of classes

//Entry point of the script
test();
```

4.6 Formatting Engine functions

The set of JavaScript classes that wrap the main classes of the AXMEDIS Format Engine includes the:

FormatManager - for managing an instance of the FormatManager and access the high level formatting functions.

ResourceList - for managing the list of resources to be formatted.

TemplateList - for managing the template library.

StyleList - for managing the style-sheet library.

Some examples of usage and scripts are also provided.

4.6.1 FormatManager

Resources management:

bool loadResourceDescriptors(string dir)
Add XML descriptors, contained in the given directory, to the list. For testing only.

ResourceList getResourceList()
Return the ResourceList object used by FormatManager.

bool clearResources()
Empty the list of resources.

int resourceNumber()
Return the number of descriptors contained in the resources list.

Profiles management:

DeviceInfo getDeviceInfo()
Return the DeviceInfo object used by FormatManager.

bool loadDeviceInfo(string filePath)
Load the XML device profile contained in the given file.

Templates management:

TemplateList getTemplateList()
Return the TemplateList object used by FormatManager.

bool loadTemplateDescriptors()
Add XML descriptors, contained in the default directory, to the list.

bool createTemplateDescriptor(string id, string location, string category, string hierarchy, string meta, string formats, string devices)
Create an XML descriptor in the default directory, using the id as filename.

bool createDescriptorsFromSMIL(string smilFile, string genTemplFile, string genStyleFile, string templateFile, string styleFile, string id)

Create a template and a style-sheet using a complete SMIL file. Also create the XML descriptors in the default directory, using the id as filename, and add them to the lists (gentemplatefile and genstylefile are the XSL files used to generate template and style-sheet).

int templateNumber()
Return the number of descriptors contained in the templates list.

Style-sheets management:

StyleList getStyleList()
Return the StyleList object used by FormatManager.

bool loadStyleDescriptors()
Add XML descriptors, contained in the default directory, to the list.

bool createStyleDescriptor(string id, string location, string tmplId, string meta, string devices)
Create an XML descriptor in the default directory, using the id as filename.

int styleNumber()
Return the number of descriptors contained in the style-sheets list.

Filtering:

string getBestTemplateId()
Return the ID of the "best" template.

string[] getBestTemplates()
Return the IDs of the automatically selected templates.

string getBestStyleId(string templId)
Return the ID of the automatically selected style-sheet, given a template.

Output management:

bool saveSMIL(string templId, string styleId, string filePath)
Create a SMIL document at the filePath, using the given template and style-sheet.

bool saveOptimizedSMIL(string templId, string styleId, string filePath, int x, int y)
Optimize (for an x/y screen) and create a SMIL document at the filePath, using the given template and style-sheet.

4.6.2 ResourceList

bool loadFile(string dir)
Load XML resource descriptors contained in the given directory.

ResourceDescriptor addResource(string resourceType, string location, string category, string parent, string related, string meta)
Create a new ResourceDescriptor and add it to the Resourcelist. Type-specific data should be entered subsequently using 'getters' methods.

int getSize()
Return the number of items contained in the list.

ResourceDescriptor getResource(int index)
Return the descriptor of the list having the given index.

string getType(int)
Return the type of the resource referred by the descriptor of the list having the given index.

4.6.3 DeviceInfo

string getDeviceClassType()

getDeviceClassType.

int getHorizontalScreensize()
getHorizontalScreensize.

int getVerticalScreensize()
getVerticalScreensize.

4.6.4 TemplateList

int getSize()
Return the number of items contained in the list.

setDescrDir(string)
Set the default directory for descriptors.

4.6.5 StyleList

int getSize()
Return the number of items contained in the list.

setDescrDir(string)
Set the default directory for descriptors.

4.6.6 Examples of usage

This script illustrates an example of usage of the classes descripted below.

```
function format()
{
    var templPath =
"../../../../Framework/doc/test/contentformatter/files/templateDescriptor";
    var stylePath =
"../../../../Framework/doc/test/contentformatter/files/styleDescriptor";
    var resPath =
"../../../../Framework/doc/test/contentformatter/files/resourceDescriptor/play-
stop";
    var devFile =
"../../../../Framework/doc/test/contentformatter/files/profiles/device1.xml";
    var outFile =
"../../../../Framework/doc/test/contentformatter/files/output/output.smi";

    var fm = new FormatManager();

    // load templates
    fm.getTemplateList().setDescrDir(templPath);
    if (fm.loadTemplateDescriptors(templPath))
    {
        if (fm.templateNumber() == 0) {
            print ("Error: template list is empty.");
            return -1;
        }
        else
            print(" - templates loaded: " + fm.templateNumber());
    }
    else {
        print("Error while loading templates.");
        return -1;
    }

    // create a new template descriptor
    if(fm.createTemplateDescriptor("play-stop",                // id
(descriptor filename)
```

```
                                    "play-stop_template.smi",    // template
filename
                                    "video-clip",                // category
                                    "",                          // hierarchy
                                    "Simple video player",       // meta
                                    "SMIL",                      // output
formats (comma separated)
                                    "PC,PDA"))                   // devices
(comma separated)
    {
        print(" - template descriptor created");
        print(" - now templates loaded are: " + fm.templateNumber());
    }
    else {
        print("Error while creating new template descriptor.");
        return -1;
    }

    // load style-sheets
    fm.getStyleList().setDescrDir(stylePath);
    if (fm.loadStyleDescriptors(stylePath))
    {
        if (fm.styleNumber() == 0) {
            print ("Error: style list is empty.");
            return -1;
        }
        else
            print(" - styles loaded: " + fm.styleNumber());
    }
     else {
        print("Error while loading styles.");
        return -1;
    }

    // create a new style-sheet descriptor
    if(fm.createStyleDescriptor("play-stop",           // id (descriptor
filename)
                                "play-stop.xsl",       // style-sheet filename
                                "play-stop",           // template id
                                0,                     // parameters
                                "Simple video player", // meta
                                "PC"))                 // devices (comma
separated)
    {
        print(" - style-sheet descriptor created");
        print(" - now style-sheets loaded are: " + fm.styleNumber());
    }
    else {
        print("Error while creating new template descriptor.");
        return -1;
    }

    // load profiles
    if (fm.loadDeviceInfo(devFile))
        print(" - device profile loaded");
    else {
        print("Error while loading profiles.");
        return -1;
    }

    // load resources
    if (fm.loadResourceDescriptors(resPath))
```

```
    {
        if (fm.resourceNumber() == 0) {
            print("Error: resource list is empty");
            return -1;
        }
        else
            print(" - resources loaded: " + fm.resourceNumber());
    }
    else {
        print("Error while loading resources.");
        return -1;
    }

    // filtering
    var tplId = fm.getBestTemplateId();
    print (" - selected template is: " + tplId);

    var stlId = fm.getBestStyleId(tplId);
    print (" - selected style-sheet is: " + stlId);

    // create output file
    if (fm.saveSMIL(tplId, stlId, outFile))
        print(" - output file created succesfully")
    else {
        print ("Error while creating output file.");
        return -1;
    }

    print(" - end");
    return 0;
}
```

4.7 Profiling Javascript Classes

The JS Profiling provides the following set of JavaScript classes:

JSAxUserProfile - Class for managing User profiles
JSAxDeviceProfile - Class for managing Device profiles
JSAxNetworkProfile - Class for managing Network profiles
Example - Example of usage and scripts

4.7.1 JSAxUserProfile

The class for managing User profiles.

Exposed Methods:

AxUserProfile()
Constructor.

boolean loadXMLFile(string url)
Loads and parses an XML profile from a file.

boolean loadXMLString(string xmlString)
Loads and parses an XML profile from an input String (in XML format).

boolean createXMLfile(string url)
Creates a new (basic) XML UserProfile, where the new file location is specified as a URI.

string getXML()
Returns the (XML formatted) contents of the XML profile in String form.

boolean setXML(string xmlString)
Write the string as the contents of the loaded XML profile, over-writing what was there previously.

string getAttribute(string domPath, string ID)
Returns the corresponding value of the specified attribute from the XML profile. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean setAttribute(string domPath, string value, string ID)
Sets a specified attribute value to the given value. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean addAttribute(string domPath, string attributeName, string attributeValue, string ID)
Adds a given attribute to the specified element. The ID parameter is optional.

boolean deleteAttribute(string domPath, string ID)
Deletes specified attribute from given element. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

string getValue(string domPath, string ID)
Returns the corresponding value of the element from the XML profile. The ID parameter is optional.

boolean setValue(string domPath, string value, string ID)
Sets the value of the specified element in the XML profile. The ID parameter is optional.

boolean createElement(string domPath, string ID)
Create an XML element using the DOM path. The new element name is specified at the end of the DOM path. The ID parameter is optional.

boolean deleteElement(string domPath, string ID)
Delete an XML element based on the DOM path. The ID parameter is optional.

boolean save()
Closes the XML profile. If any element is changed then the changes are saved first.

4.7.2 JSAxDeviceProfile

The class for managing Device profiles.

Exposed Methods:

AxUserProfile()
Constructor.

boolean loadXMLFile(string url)
Loads and parses an XML profile from a file.

boolean loadXMLString(string xmlString)
Loads and parses an XML profile from an input String (in XML format).

boolean createXMLfile(string url)
Creates a new (basic) XML DeviceProfile, where the new file location is specified as a URI.

string getXML()
Returns the (XML formatted) contents of the XML profile in String form.

boolean setXML(string xmlString)

Write the string as the contents of the loaded XML profile, over-writing what was there previously.

string getAttribute(string domPath, string ID)
Returns the corresponding value of the specified attribute from the XML profile. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean setAttribute(string domPath, string value, string ID)
Sets a specified attribute value to the given value. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean addAttribute(string domPath, string attributeName, string attributeValue, string ID)
Adds a given attribute to the specified element. The ID parameter is optional.

boolean deleteAttribute(string domPath, string ID)
Deletes specified attribute from given element. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

string getValue(string domPath, string ID)
Returns the corresponding value of the element from the XML profile. The ID parameter is optional.

boolean setValue(string domPath, string value, string ID)
Sets the value of the specified element in the XML profile. The ID parameter is optional.

boolean createElement(string domPath, string ID)
Create an XML element using the DOM path. The new element name is specified at the end of the DOM path. The ID parameter is optional.

boolean deleteElement(string domPath, string ID)
Delete an XML element based on the DOM path. The ID parameter is optional.

boolean save()
Closes the XML profile. If any element is changed then the changes are saved first.

4.7.3 JSAxNetworkProfile

The class for managing Network profiles.

Exposed Methods:

AxUserProfile()
Constructor.

boolean loadXMLFile(string url)
Loads and parses an XML profile from a file.

boolean loadXMLString(string xmlString)
Loads and parses an XML profile from an input String (in XML format).

boolean createXMLfile(string url)
Creates a new (basic) XML NetworkProfile, where the new file location is specified as a URI.

string getXML()
Returns the (XML formatted) contents of the XML profile in String form.

boolean setXML(string xmlString)
Write the string as the contents of the loaded XML profile, over-writing what was there previously.

string getAttribute(string domPath, string ID)
Returns the corresponding value of the specified attribute from the XML profile. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean setAttribute(string domPath, string value, string ID)
Sets a specified attribute value to the given value. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

boolean addAttribute(string domPath, string attributeName, string attributeValue, string ID)
Adds a given attribute to the specified element. The ID parameter is optional.

boolean deleteAttribute(string domPath, string ID)
Deletes specified attribute from given element. The attribute name is specified at the end of the DOM path. The ID parameter is optional.

string getValue(string domPath, string ID)
Returns the corresponding value of the element from the XML profile. The ID parameter is optional.

boolean setValue(string domPath, string value, string ID)
Sets the value of the specified element in the XML profile. The ID parameter is optional.

boolean createElement(string domPath, string ID)
Create an XML element using the DOM path. The new element name is specified at the end of the DOM path. The ID parameter is optional.

boolean deleteElement(string domPath, string ID)
Delete an XML element based on the DOM path. The ID parameter is optional.

boolean save()
Closes the XML profile. If any element is changed then the changes are saved first.

4.7.4 Examples of usage

```
var userprof = new AxUserProfile();
userprof.loadXMLFile("userprofile.xml");
userprof.setValue("Preferences/AudioPreferences/VolumeControl","0.90",
"ID000000");
userprof.save();
```

4.8 Metadata Mapper Javascript Class

The set of JavaScript classes that wrap the main classes for addapting AXMEDIS Metadata.

List of classes and link to internal sections

Metadata Mapper - Class for Adapting Metadata using an XSLT
Example of usage

4.8.1 MetaDataMapper

Description

Exposed methods:

MetaDataMapper()
Constructor

*string ExtractMetadata(AxObjectManager * axom)*
Method to extract Metadata from an AXMEDIS Object returning the result as a standard string with XML.
AxObjectManager pointer to an instance of an AXMEDIS Object

*bool EmbedMetadata(AxObjectManager * axom, string xmlMetadata)*
Method to embed Metadata into an AXMEDIS Object returning the result true if successful.
AxObjectManager pointer to an instance of an AXMEDIS Object, standard string of metadata XML to

embed

*void LoadXML(AxObjectManager * axom, string xmlMetadata)*
Method to load the XSLT file with no return. Input paramters are a standard string to locate the .xsl file.

string Transform(string xmlMetadata)
Method to Transform XML Metadata string into a new Metadata XML string returning the standard string of the new XML Metadata. Standard string pointer of the XML Metadata string to be converted.

4.8.2 Examples of usage

The following script loads an AXMEDIS Object from a local file source, initialises the MetaDataMapper and extracts the metadata into an XML string. The script loads the XSLT (.xsl) from a local file source and uses this to transcode the metadata string into a new adapted metadata XML string. the string is embedded into the original AXMEDIS object and saved with a new file name.

```
function test()
{
   // Open AXMEDIS object
   var axom = new AxmedisObject("C:\\axmedis\\tiscali-mediaclub-dottor-
mabuse.axm");

   // Initalise the Metadata Mapper
   var mdm = new MetaDataMapper();

   // Extract the Metadata from the AXMEDIS object
   var extractedString = mdm.ExtractMetadata(axom);

   // Load the saved xslt file
   mdm.LoadXSL("C:\\axmedis\\tiscali-test-style2.xsl");

   // Transform the metadata returning the new metadata string
   var out = mdm.Transform(extractedString);

   // Embed the new metadata into an axmedis object
   mdm.EmbedMetadata(axom, out);

   // Save AXMEDIS object
   axom.save("C:\\axmedis\\tiscali-mediaclub-dottor-mabuse-adapted.axm")

   return 0;
}
//Entry point of the script
test();
```

4.9 AXEPTool Javascript Classes

The AXEPTool provides the following set of JavaScript classes.

List of classes and link to internal sections
AXP2PManager - Class for managing calls to an AXEPTool WS.
DownloadedObjectListResponse - Class for managing a downloaded object list response from AXEPTool WS.
DownloadStatusResponse - Class for managing an object download status response from AXEPTool WS.
PublicationStatusResponse - Class for managing a publication status response from AXEPTool WS.
PublishedObjectListResponse - Class for managing a published object list response from AXEPTool WS.
Example of usage

4.9.1 AXP2PManager

This Class manages calls to an AXEPTool WS.

Exposed properties:

string endpointUri
The AXEPTool's Uri.

string trackerUri
The Tracker's Uri.

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.

boolean getLastError()
Gets the last gsoap error.

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 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 Uri, which can be its AXOID or local file name) 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.

4.9.2 DownloadedObjectListResponse

This Class manages the downloaded object list response.

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).

4.9.3 DownloadStatusResponse

This Class manages the Downloaded Object Status response.

Exposed properties:

int nseeds
Number of seeds.

int dkbytes
Amount of downloaded KBytes.

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 getdKBytes()
Gets the downloaded KBytes.

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.

4.9.4 PublicationStatusResponse

This Class manages the Publication Object Status response.

Exposed properties:

int numberOfSeeds
The number of seeds.

int uploadedKBytes
The amount of uploaded KBytes.

array PeersList
The peers list.

Exposed methods:

int getnseeds()
Gets the number of seeds.

int getkbytes()
Gets the amount of uploaded KBytes.

array getPeersList()
Gets the peers list.

4.9.5 PublishedObjectListResponse

This Class manages the Published Object List response

Exposed properties:

array publishedObjectList
The list of published 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 published object referred by *index*. If index is greater than the size of the published object list the method returns an empty string.

string getinfoHash(int index)
Gets the infoHash of the published object referred by *index*. If index is greater than the size of the published object list the method returns an empty string.

string getlocalFileName(int index)
Gets the local file name of the published object referred by *index*. If index is greater than the size of the published object list the method returns an empty string.

array getPublishedObjectList()
Gets the Published Object List as a matrix of strings where each row provides values for *AXOID* (column 0), *InfoHash* (column 1) and *LocalFilename* (column 2).

4.9.6 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");
```

4.10 AXMEDIS JAVASCRIPT Functions

The set of JavaScript functions prthat wraps the main classes of the AXMEDIS Object Model includes the:

I/O functions - for managing an instance of the AXMEDIS Object
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
Example - Example of usage and scripts

4.10.1 I/O functions

print(string msg)
Print a string on the current I/O device. It could be the GUI, the console or other.

4.10.2 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.

4.10.3 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

boolean changeDir(string dirPath)
Command to change the folder path into the filesystem returning TRUE in the success case.

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 "*.*"

4.10.4 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.

4.10.5 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

4.10.6 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 );
    }
}
The following script shows how to browse directories in the unit C:
var l = listDir("c:\\"); //It provides the list of folder in c:
for (i in l)
{
    print(l[i]);
    var f = getFirstFile(l[i]);
    while(f!=null){
        print(f); // here, operations to do with the file name
        f = getNextFile();
    }
}
```

4.11 Utility and other Javascript Classes

The searchbox bridge provides the following set of JavaScript classes:

ZipArchiver - Class for managing archives (zip/unzip)
Example - Example of usage and scripts

4.11.1 ZipArchiver

The class for managing archive of file (zip/unzip).

Exposed Properties:

zipFile: string
The string contains the file as array of chars

Exposed Methods:

ZipArchiver()
It creates a ZipArchiver object. Constructor.

boolean createNewZipArchive(string filePath)
It creates a new zip file at the filePath location. The filePath has to contain the name of file.

boolean openZipArchive(string filePath)
It opens a existing zip archive at the filePath location.

boolean addFileToZip(string zipFilePath,string filePath)
It adds a file at the filePath location to the zipFilePath archive.

boolean 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.

4.11.2 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();
```

5 E4X Short Guide for Javascript v. 1.6

ECMAScript for XML (**E4X**) is a programming language extension that adds native XML support to JavaScript (ver. 1.6). It does this by providing access to the XML document in a form that feels natural for ECMAScript programmers. The goal is to provide an alternative, simpler syntax for accessing XML documents than via DOM interfaces.

E4X is standardized by Ecma International in ECMA-357 standard (currently in its first edition, June 2004).

E4X is implemented (at least partially) in SpiderMonkey

5.1 Known bugs and limitations

- It is not currently possible to access a DOM object through E4X (bug 270553)
- E4X doesn't support parsing XML declaration (`<?xml version=...?>`) (bug 336551). You may get SyntaxError "xml is a reserved identifier" (despite the XML being in a string).

Workaround:

```
var response = xmlhttprequest.responseText; // bug 270553
response = response.replace('<?xml version="1.0"?>', ""); // bug 336551
var e4x = new XML(response);
```

5.2 Resources

- See the list of E4X-related pages on MDC.
- ECMA-357 standard
- E4X at faqts.com

5.3 E4X - Keywords, Statements, Identifiers and Punctuators

5.3.1 Context Keywords

E4X adds three context keywords (each, XML, namespace). Context keywords take on a specific meaning when used in specified contexts where identifiers are not permitted by the syntactic grammar.

5.3.2 Statements

The **default XML namespace** of the global scope has the initial value of no namespace. You can define its scope also. For example, if you define it within a function call, the value set will not be accessible outside the function. To deal with XML with namespaces easily, you can declare a default namespace for the current scope (e.g., function scope or global scope) using the newly introduced statement, as follows:

```
default xml namespace = expression;
```

The above statement should evaluate to a string with a URI or to a namespace object. The result is that all elements created in the current scope will be in the default namespace, provided they have no prefix in their name. The following code snippet shows the uri value when the default namespace is both defined and not defined:

```
var xmlElement1 = <p>Hello how are you</p>;
// shows '';
alert(xmlElement1.name().uri);

// set default namespace
default xml namespace = 'http://www.w3.org/1999/xhtml';
var xmlElement2 = <p>Hello how are you</p>;
```

```
 // shows 'http://www.w3.org/1999/xhtml';
  alert(xmlElement2.name().uri);
```

The **for-in statement** evaluates an expression and iterates through each property of the resulting XML object. When the expression evaluates to an XML object, the for-in statement converts the XML object to an XMLList and iterates over the resulting XMLList. The following snippet shows the usage of a for-in loop statement:

```
var xmlElement = <publishing>
                    <author type="freelance" books="java">Rahul Gupta</author>
                    <author type="employed">Jon</author>
                   </publishing>;

//show  Rahul Gupta, freelance, Jon, employed
for (var childElement in xmlElement.author)
  {
     alert("Element No ["+childElement +"] = " +
xmlElement.author[childElement]);
     for (var childAttribute in xmlElement.author[childElement].@type)
        alert("Attribute No ["+childAttribute +"] = " +
xmlElement.author[childElement].@type);
  }
```

The **for-each-in statement** iterates through each property of the resulting XMLList object in order. The following code snippet shows the usage of for-each-in statement:

```
var xmlElement = <publishing>
 <author type="freelance" books="java">Rahul Gupta</author>
<author type="employed">Jon</author>
</publishing>;

//shows  Rahul Gupta, freelance, Jon, employed
for each (var xmlElements in xmlElement.author) {
    alert( " Element = "+ xmlElements);
    for each (var xmlAttributes in xmlElements.@type)
    alert( " Attribute = "+ xmlAttributes.toXMLString());
 }
```

The for-each-in statement behaves differently from the for-in statement. In particular, it assigns the loop variable over the range of the object rather than the domain of the object. For example, the for-each-in statement binds the loop variable to the property values of the given object rather than the property names.

5.3.3 Identifiers and Punctuators

Table 1 lists E4X identifiers and punctuators with descriptions and accompanying examples.

Identifiers	Sign/Represented By	Description	Example

Attribute Identifiers	**@**	An attribute identifier is used to identify the name of a XML attribute. It evaluates to a value of type AttributeName. The preceding "@" character distinguishes an XML attribute from an XML element with the same name.	`var xmlElement = <publishing>`
			`<author type="freelance">Rahul`
			`Gupta</author>`
			`</publishing>;`
			`// shows 'freelance'`
			`alert(xmlElement.author.@type);`

Qualified Identifiers	**::**	A qualified identifier is used to identify values defined within a specific namespace. They may be used to access, manipulate, and create namespace-qualified XML elements and attribute names.	`var xmlElement3 = <html`
			`xmlns="http://www.w3.org/1999/xhtml"`
			`xml:lang="en">`
			`<body>`
			`<a href="www.google.com"> Google </a>`
			`</body> </html>`
			`var xhtml = new`
			`Namespace('http://www.w3.org/1999/xhtml');`
			`var xhtmlElements = xmlElement3.xhtml::*;`
			
			`//shows 1 - for body`
			`alert(xhtmlElements.length());`

Wildcard Identifiers	*****	A wildcard identifier is used to identify any name. It may be used for matching namespaces, properties of XML objects, or XML attributes.	`var xmlElement = <publishing>`
			`<author type="freelance"`
			`books="java">Rahul</author>`
			`<author type="employed">Jon</author>`
			`</publishing>;`
			`//shows all elements and Attributes`
			`for each (var xmlElements in xmlElement.*)`
			`{`
			`alert("Element="+ xmlElements);`
			`for each (var xmlAttributes in`
			`xmlElements.@*)`
			`alert("Attribute="+`
			`xmlAttributes.toXMLString());`
			`}`

Descendent Accessor	**..**	A descendent accessor supports the XML descendent (Child, grandchild, great-grandchild) accessor.	`var xmlElement =`
			`<publishing>`
			`<author type="freelance">Rahul`
			`Gupta</author>`
			`<author type="employed">Jon</author>`
			`</publishing>;`
			`// shows 'freelanceemployed'`
			`alert(xmlElement..@type);`

Table 1. E4X Identifiers and Punctuators

5.4 E4X - Operators & Objects

5.4.1 Operators

Table 2 lists E4X operators with descriptions and accompanying examples.

Operator	Sign/Represented By	Description	Example
Delete Operator	"delete" keyword	This operator is for deleting XML properties and XML attributes from XML objects and XMLLists.	`var xmlElement = <publishing> <author type="freelance" books="java">Rahul Gupta</author> </publishing>; delete xmlElement.author.@books; // will not show "java" attribute for each (var xmlAttributes in xmlElement..@*) alert("Attribute="+ xmlAttributes.toXMLString());`
Typeof Operator	"typeof" keyword	This operator is for determining the types of XML and XMLList objects.	`var xmlElement = <author type="freelance" >Rahul Kumar Gupta</author>; // show "xml" alert(typeof(xmlElement));`
Addition Operator	+	Depending on its arguments, this operator is for performing string concatenation, XML and XMLList concatenation, or numeric addition.	`var xmlElement = <publishing> <book price="100">Java</book> <book price="150">JSP</book> </publishing>; totPrice= +xmlElement.book[0].@price + +xmlElement.book[1].@price; // show "250" alert(totPrice);`
Equality Operator	==	This operator is for enabling equality comparisons (including E4X objects like QName and Namespace objects, and the types XML and XMLList).	`var xmlElement1 = <book price="100">Java</book>; var xmlElement2 = <book price="150">Java</book>; // shows "false" alert( (xmlElement1 == xmlElement2));`
Assignment Operator	=	This operator is used to modify, replace, and insert properties and XML attributes in an XML object. It has two types: 1. XML Assignment 2. XMLList Assignment	`var xmlElement = <publishing> <book price="100">Java</book> <book price="150">JSP</book> </publishing>; xmlElement.book[1].@price = 165; // shows "165" alert(xmlElement.book[1].@price);`

Compound Assignment	op=	E4X benefits from the compound assignment operator "+=". It is used for inserting the XML objects specified by the AssignmentExpression just after the XML objects specified by the LeftHandSideExpression in the context of their parent.	`var xmlElement = <publishing> <book price="100">Java</book> <book price="150">JSP</book> </publishing>; xmlElement.book[0] += <book price="200">EJB</book>; //shows JAVA. EJB JSP for each (var xmlElements in xmlElement.*) alert("Element="+ xmlElements);`
XML Filtering Predicate Operator	xml(list)Object.(filtering predicate expression)	This operator is used to apply a condition to filter out certain elements in a child node or descendent accessor.	`var xmlElement= <publishing> <author type="freelance">Rahul</author> <book price="200">EJB</book> <book price="150">JSP</book> </publishing>; //show EJB for each (var xmlElements in xmlElement.*) { if(xmlElements.@price == '200') alert(xmlElements);`

Table 2. E4X Operators

5.4.2 E4X Objects

E4X adds four native objects to ECMAScript:

- Namespace
- QName
- XML
- XMLList

In addition, E4X adds new properties to the global object. It adds native XML support to the language, meaning in addition to types like Number, String, Boolean, and Object, there is the XML type for representing XML elements, attributes, comments, processing-instructions, or text nodes, and the XMLList type for representing a list of XML objects. Both XML and XMLList are new fundamental data types for representing XML objects and lists of XML objects.

The **XML type** is an ordered collection of properties with a name, a set of XML attributes, a set of in-scope namespaces, and a parent. The following is an example of an XML object referring to one root node:

```
<rootnode>
<node></node>
<node></node>
<node></node>
</rootnode>
```

An **XML initializer** is an expression written in a form of a literal that describes the initialization of an XML object. It may specify a XML element, a XML comment, a XML PI, or a CDATA section using ordinary XML syntax. The following sample code shows one:

```
// an XML object representing a author with a name and age
var author = <author> <name>Rahul Gupta</name>
<authorType>freelance</authorType></author>;
```

```
//on XML Object representing two authors
var publishing = <publishing>
 <author type="freelance" books="java"> <name>Rahul Gupta</name>  </author>
  <author type="employed"> <name>Jon</name> </author>
</publishing>;
```

The **XMLList type** is an ordered collection of properties. A value of type XMLList represents a XML document, an XML fragment, or an arbitrary collection of XML objects. This is an example of an XMLList that refers to multiple nodes:

```
<node></node>
<node></node>
<node></node>
```

A XMLList initializer is an expression describing the initialization of an XMLList object. It describes an ordered list of XML properties using an anonymous XML element syntax. XMLList initializers begin with the character sequence "<>" and end with the character sequence "</>". The following snippet shows a way to initialize the XMLList object in E4X code:

```
var xmlIstObj = <><name>Rahul Gupta</name>
<authorType>freelance</authorType></>;
var authorlistObt = <>
  <author type="freelance" books="java">  <name>Rahul Gupta</name>  </author>
  <author type="employed"> <name>Jon</name> </author>
</>;
```

The main difference between the two is that with XML you are dealing with one specific object that can contain n number of child nodes while with XMLList you are dealing with a collection of one or more XML objects.

To simplify the programmer's task, E4X intentionally blurs the distinction between a single XML object and an XMLList containing only one value. To that end, E4X extends the ECMAScript's function call semantics such that all methods available for values of type XML are also available for XMLLists of size one.

Table 3 lists E4X methods and properties with descriptions and accompanying examples.

Methods/Properties	Available in XML	Available in XML List	Description
addNamespace(..)	Y		The addNamespace method adds a namespace declaration to the in-scope namespaces for an XML object and returns the XML object.
appendChild(..)	Y		The appendChild method appends the given child to the end of an XML object's properties and returns the XML object. ***Alternatives:*** 1. *Use the insertChildBefore method with the first parameter value as null.* 2. *Use a compound operator.*
attribute(..)	Y	Y	The attribute method takes the names of the attributes to look for. ***Alternative:*** *xml.@attributeName* OR *xml.['@attributeName']*

attributes(..)	Y	Y	The attributes method returns a XMLList containing the XML attributes. ***Alternative:*** *xml.@** OR *xml ['@*']*
child(..)	Y	Y	The child property takes the child property name to look for. ***Alternative:*** *xml.ChildPropertyName* OR *xml['ChildPropertyName']*
childIndex(..)	Y		The childIndex method returns a number representing the ordinal position of an XML object within the context of its parent. When xmlobject has no parent, the special number value NaN (not a number) is returned.
children(..)	Y	Y	The children method returns a XMLList containing all the child properties. ***Alternative:*** *use wild card **
comments (..)	Y	Y	The comments method returns an XMLList containing the properties that represent XML comments. Make sure that you have the *XML.ignoreComments = false;* setting before creating XMLobject.
contains(..)	Y	Y	The contains method takes one argument: the value to check for in xmlobject/XmlList.
copy(..)	Y	Y	The copy method returns a deep copy of this XML object/XmlList. You can make a copy of either all or part of XML object/XmlList.
descendants (..)	Y	Y	The descendants method returns an XMLList with thedescendants matching the passed name argument or with all descendants, if no argument is passed. ***Alternative:*** *Use the descendants operator (..)*
elements(..)	Y	Y	The elements method takes on a parameter and returns an XMLList with all matching child elements or with all child elements if no argument was passed. ***Alternative:*** `xml.childElementName xml ['childElementName']`
hasOwnProperty(..)	Y	Y	The hasOwnProperty method returns a Boolean value indicating whether the object/List has the property specified by parameter.
hasComplexContent(..)	Y	Y	The hasComplexContent method returns a Boolean value indicating whether the XML object/XMLList contains complex content (i.e., an XML element that has child elements).
hasSimpleContent(..)	Y	Y	The hasSimpleContent method returns a Boolean value indicating whether the XML object/list contains simple content (i.e., a text node, attribute node) or represents an XML element that has no child elements. ***Note:*** *The existence of attributes, comments, processing instructions, and text nodes within an XML object/list is not significant for determining if it has simple content.*

isScopeNamespaces(..)	Y		The isScopeNamespaces method returns an array of Namespace objects representing the namespaces that are in scope for this XML object.
insertChildAfter(...)	Y		The insertChildAfter method takes two arguments (p1 and p2), an existing child (p1) to insert after and the new child (p2) to be inserted. If p is null, it inserts p2 before all the children of an XML object. If p1 does not exist in this XML object, it returns without modifying the XML object.
insertChildBefore(..)	Y		The insertChildBefore method takes two arguments (p1 and p2), an existing child (p1) to insert before and the new child (p2) to be inserted. If p1 is null, it inserts p2 after all children in the XML object. If p1 does not exist in the XML object, it returns without modifying the XML object.
length(..)	Y	Y	The length method returns the number of XMLObjects the XMLList contains. ***Note:*** *For XmlObject, it always returns 1.This is to make a distinction between an XML object and an XMLList object containing exactly one value.*
localName(..)	Y		The localName method returns the local name portion of the qualified name of the XML object. ***Note:*** *Text nodes and comment nodes do not have a name, thus for these kinds of XML objects, null is returned.*
name(..)	Y		The name method returns the qualified name associated with the XML object or null if the object has no name (as for text nodes and comment nodes).
namespace(..)	Y		The namespace method takes one optional parameter (p1). If p1 is not specified, it returns the namespace associated with the qualified name of the XML object. Otherwise, it looks for a matching namespace in the in-scope namespace of the XML object and returns it.
namespaceDeclarations(..)	Y		The namespaceDeclarations method returns an array of Namespace objects representing the namespace declarations associated with the XML object in the context of its parent.
nodeKind(..)	Y		The nodeKind method returns a string representing the type of node the XMLObject is. The possible values are element, attribute, text, comment, and processing-instruction.
normalize(..)	Y	Y	The normalize method returns XMLObject after joining adjacent text nodes and eliminating empty text nodes.
parent(..)	Y	Y	The parent method returns the parent of this XML object, or null if there is no parent. ***Note:*** *The parent of the attribute is an element to which the attribute is attached.*

processingInstructions(..)	Y	Y	The processingInstructions method takes one parameter, the PI name to look for, and returns the XMLList with all the matching PI child nodes. ***Note:*** *If no parameter is defined, then the wild card name '*' is used to match all PI.*
prependChild(..)	Y		The prependChild method inserts a specified child as the first child in xmlObject. ***Alternatives:*** *1. Use the insertChildAfter method with the first parameter value as null. 2. Use a compound operator.*
propertyIsEnumerable(..)	Y	Y	The propertyIsEnumerable method determines whether the specified property will be included in the set of properties iterated over when the XML object is used in a for-in statement. The method returns **true** when ToString(P) is **"0"**; otherwise **false**.
removeNamespaces(..)	Y		The removeNamespaces method removes the namespace from the in-scope namespaces of the XML object.
replace(..)	Y		The replace method takes two parameters (p1 and p2) and replaces the XML properties of the XML object specified by propertyName (p1) with value (p2) and returns this XML object. ***Note:*** *The propertyName parameter may be a numeric property name, an unqualified name for a set of XML elements, a qualified name for a set of XML elements, or the properties wildcard "*".*
setChildren(..)	Y		The setChildren method takes two parameter (p1 and p2). It replaces the XML properties (p1) of the XML object with a new set of XML properties from *value* (p2). *Value* may be a single XML object or an XMLList. SetChildren returns this XML object.
setLocalName(..)	Y		The setLocalName method changes the local name of this XMLObject with the name passed in.
setName(..)	Y		The setName method changes the QName of the XMLObject with the name passed in. The parameter can be a string or a QName object.
setNamespace(..)	Y		The setNamespace method changes the namespace associated with the name of the XML object to the new specified namespace.
text(..)	Y	Y	The text method returns an XMLList containing all XML properties of the XML object that represent XML text nodes.
toString(..)	Y	Y	The toString method returns a convenient string representation of the XMLList/XmlObject. ***Note:*** *If you are looking for complete serialization of all XML objects in the XMLList, use toXMLString.*

toXMLString(..)	Y	Y	The toXMLString method returns an XML-encoded string representation of the XMLList/XMLObject. It always includes the start tag, attributes, and end tag of the XML object regardless of its content. Global settings such as XML.prettyPrinting and XML.prettyIndent format and indent the resultant string.
valueOf(..)	Y	Y	The valueOf method simply returns the XML object/XML list it is called on.

Table 3. E4X Methods and Properties

QNames (Qualified Names) represent qualified names of XML elements and attributes. Each QName object has a local name of type string and a namespace URI of type string or null. When the namespace URI is null, the qualified name matches any namespace. A QName object can be created by using the following:

```
QName ( ) or QName ( Name ) or QName (Namespace,  Name )

              var Qnamevar = new QName('http://www.w3.org/1999/xhtml',
'author');
```

An XML object representing an element or attribute has an internal name property, which is a QName object having two properties, localName and uri. You can get the QName of any XMLObject by using the name(..) method. Table 4 lists QName methods and properties with their descriptions and accompanying examples.

Methods/Properties	Description	Example

localName	The value of the localName property is a value of type string. When the value of the localName property is "*", it represents a wildcard that matches any name.	`QNameVar =`
		
		`QName("http://www.w3.org/1999/xhtml","author");`
		`//shows author`
		`alert(QNameVar.localName);`

uri	The value of the uri property is null or a value of type string identifying the namespace of the QName. When the value of the uri property is the empty string, this QName is said to be in no namespace.	`QNameVar =`
		
		`QName("http://www.w3.org/1999/xhtml","author");`
		`//shows http://www.w3.org/1999/xhtml`
		`alert(QNameVar.uri);`

Table 4. QName Methods and Properties

Namespace objects represent XML namespaces and provide an association between a namespace prefix and a unique resource identifier (URI). The prefix is either the undefined value or a string value that may be used to reference the namespace within the lexical representation of an XML value.
You can create a Namespace object using either of the following:

```
Namespace ( )  or   Namespace ( uriValue ) or Namespace ( prefixValue, uriValue
```

```
)
       var namespaceVar = new
Namespace("author",'http://www.w3.org/1999/xhtml');
```

Table 5 lists namespace methods and properties with their descriptions and accompanying examples.

Methods/Properties	Description	Example

prefix	The value of the *prefix* property is either the **undefined** value or a string value.	`var xmlElement = <xhtml:p xmlns:xhtml="http://www.w3.org/1999/xhtml">Hello how are you.</xhtml:p>;`
		`var namespace = xmlElement.namespace();`
		`//shows 'xhtml'`
		`alert(xmlElement.namespace().prefix);`

uri	The value of the *uri* property is a string value. When the value of the *uri* property is the empty string, the Namespace represents the *unnamed namespace* in no namespace.	`var xmlElement = <p>Hello how are you</p>;`
		`// shows ''`
		`alert(xmlElement.name().uri);`

Table 5. Namespace Methods and Properties

When no namespace is defined for an element/XML, the uri property has as its value an empty string. Otherwise, the uri property has as its value the namespace URI. The following code shows the usage of the uri method:

```
var xmlElement = <p>Hello how are you</p>;
 // shows ''
 alert(xmlElement.name().uri);
 xmlElement = <p xmlns="http://www.w3.org/1999/xhtml"> Hello how are you.</p>;

  // shows 'http://www.w3.org/1999/xhtml
alert(xmlElement.name().uri);
```

5.5 E4X - Global Methods and Properties

5.5.1 Global Methods and Properties

Table 6 lists namespace global methods and properties with their descriptions and accompanying examples.

Methods/Properties	Description	Example

ignoreComments	If ignoreComments is true, XML comments are ignored when constructing new XML objects.
	
	Default -true
		`XML.ignoreComments = false;`
		`var xmlElement = <author> <!-- info line --> Rahul Gupta </author>;`
		`for each (var xmlElements in xmlElement.*)`
		`alert(" Element = "+ xmlElements);`

ignoreProcessingInstructions	If ignoreProcessingInstructions is true, XML-processing instructions are ignored when constructing new XML objects.
	
	Default -true
		`XML.ignoreProcessingInstructions= false;`
		`var xmlElement = <publishing>`
		`<?process author="yes"?>`
		`<author type="freelance">Rahul Gupta </author>`
		`</publishing>;`
		`for each (var xmlElements in xmlElement.*)`
		`alert(" Element = "+ xmlElements);`

ignoreWhiteSpace	If ignoreWhitespace is true, insignificant whitespace characters are ignored when processing or constructing new XML objects. Whitespace characters are defined to be space (\u0020), carriage return (\u000D), line feed (\u000A), and tab (\u0009).
	
	Default -true
		`XML.ignoreWhitespace = false;`
		`var xmlElement =`
		`<publishing>`
		`<author>Rahul Gupta</author>`
		`</publishing>;`
		`alert(xmlElement.toString());`

prettyIndent	Child nodes will be indented relative to their parent node by the number of spaces specified by prettyIndent. This is effective only when prettyPrinting is true.
	
	Default -2
		`alert(xmlElement.toXMLString());`
		
		`XML.prettyPrinting=true;`
		`XML.prettyIndent = 4;`
		`var xmlElement =`
		`XML.prettyPrinting=false;`

prettyPrinting	If prettyPrinting is true, the ToString and ToXMLString operators will normalize whitespace characters between certain tags to achieve a uniform appearance.
	
	Default -true
		`var xmlElement = <publishing><author>Rahul Gupta </author>`
		`<author>Jon </author></publishing>;`
		
		`alert(xmlElement.toXMLString());`
		
		`XML.prettyPrinting=true;`
		`XML.prettyIndent = 4;`
		`var xmlElement =`
		`<publishing><author>Rahul Gupta </author>`
		`<author>Jon </author></publishing>;`
		
		`alert(xmlElement.toXMLString());`

isXMLName(..)	IsXMLName is used to determine whether it is a valid XML name that can be used as an element or attribute name.	`//shows false`
		`alert(isXMLName("?xml"));`
		`//shows true`
		`alert(isXMLName("author"));`

Table 6. Namespace Global Methods and Properties

5.5.2 Advantages and Limitations of E4X

One more ECMAScript technique for handling XML is object mapping. You can map XML data onto a set of ECMAScript objects, manipulate those objects directly, and then map them back to XML. This approach has the advantage of allowing ECMAScript programmers to use their existing knowledge, but it has some problems. For example, an ECMA native object can't preserve the order of the original XML data. As E4X evolves, this approach will mature and become standardized. Table 7 summarizes the advantages and limitations of E4X.

Area	Description
Browser Support	As of now (Nov. 2006), Mozilla, FireFox 1.1 +, and Safari provide limited support. IE7 doesn't support E4X, but it is becoming standard so in future all browser should support it.
Code Size	In general, the size of the code you produce when using E4X for a requirement tends to be smaller than the code generated with other XML-related technologies.
Learning Curve	The learning curve is very low, as it is an extension to JavaScript.
Cost	Both development and maintenance costs are lower than other related technologies such as XSLT, XML queries, or DOM routines.
Performance	E4X is lighter weight than similar technologies so it has better performance.

Table 7. Advantages and Limitations of E4X

5.5.3 E4X by Example

Create a literal document:

```
doc = <a><b c="1"/><b c="2"/></a>;
```

Create cursor into document:

```
b0 = doc.b[0];
```

Add/assign an attribute:

```
doc.@c="new value";
```

Delete a node:

```
delete doc.@c;
```

Iterate over nodes:

```
for each (b in doc.b) { x = b.@c; }
```

Find all decendants:

```
allc = doc..@c;
```

Iinsert a new fragment:

```
frag = <b c="3"/>;
doc.insertChildAfter(doc.b[1], frag);
```

Iinsert a new fragment at beginning:

```
frag = <b c="0"/>;
doc.insertChildAfter(null, frag);
```

Parameterized locate:

```
cValue=2;
bNode = doc.b.(@c==cValue);
```

Parameterized text in literal:

```
text="text here";
frag = <b c="4">{text}</b>;
```

Parameterized attribute in literal:

```
text="4";
frag = <b/>;
frag.@c=text;
```

Count matches:

```
count = doc.b.length();
```

6 AXMEDIS Content Processing Functionalities in Plug-ins

6.1 Audio Adaptation plug-in

Category: ContentProcessing
Identifier: AudioAdaptation
Library: AudioAdaptation Version: 1.001
Vendor: Axmedis
Main Library: audioadaptationplugin.dll

Description: Plugin for audio processing

FunctionList:
* FFAudioTranscoding
* LSAudioTranscoding

6.1.1 FFAudioTranscoding

STRING FFAudioTranscoding (RESOURCE InputResource, STRING Mimetype, RESOURCE OutputResource, UINT32 OutputSamplingRate, UINT16 OutputNumChannels, UINT16 OutputBitRate, FLOAT ReadStartingTime, FLOAT ReadEndingTime, STRING OutputCodec)

Version: 1.0
Description: Transcoding of audio files by FFMPEG
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-ms-wma basic x-mpeg x-vorbis x-pn-realaudio x-ac3 x-dv x-mace x-adpcm x-aac 32KADPCM amr x-pn-realaudio video
Resource Format: x-mpeg x-mpeg2 mp4 x-raw x-h263 x-mjpeg x-ms-wmv x-ms-asf x-flv x-svq x-dv x-h264 x-indeo x-vp3 x-ffv x-vcr x-msvideo x-nut application
Resource Format: x-pcm vnd.rn-realmedia x-shockwave-flash
Ranges:
Name: Mimetype
Description: Mimetype for output resource
Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: OutputSamplingRate
Description: Sampling rate of the output audio file (default: sampling rate of the input)
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputNumChannels

Description: Number of channels of the output audio file (default: number of channels of the input)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputBitRate
Description: Bit rate of the output audio file - Only applies to compressed audio file formats (default: 64 kb)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: ReadStartingTime
Description: Starting time in the input audio file (default: beginning of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: ReadEndingTime
Description: Ending time in the input audio file (default: end of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputCodec
Description: Codec of the output audio file (default: depends on the desired format of the output)
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.1.2 LSAudioTranscoding

STRING LSAudioTranscoding (RESOURCE InputResource, STRING Mimetype, RESOURCE OutputResource, FLOAT ReadStartingTime, FLOAT ReadEndingTime, STRING OutputCodec)

Version: 1.0
Description: Transcoding of audio files by LibSnd
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-aiff basic x-paris x-svx x-nist x-voc x-ircam x-w64 x-sd2 x-flac application

Resource Format: x-pcm x-pagerecall
Ranges:
Name: Mimetype
Description: Mimetype for output resource
Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: ReadStartingTime
Description: Starting time in the input audio file (default: beginning of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: ReadEndingTime
Description: Ending time in the input audio file (default: end of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputCodec
Description: Codec of the output audio file (default: depends on the desired format of the output)
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.2 Audio Descriptor plug-in

Category: ContentProcessing
Identifier: AudioDescriptor
Library: AudioDescriptor **Version:** 1.001
Vendor: Axmedis
Main Library: audiodescriptorplugin.dll

Description: Plug-in for audio descriptors extraction

FunctionList:
* LowLevelDescriptors
* Segmentation
* TempoEstimation
* MusicGenreEstimation

6.2.1 LowLevelDescriptors

STRING LowLevelDescriptors (RESOURCE InputResource, FLOAT HopSize, UINT16 AudioPower, UINT16 SpectralCentroid, UINT16 SpectralSpread, UINT16 SpectralEnvelope, FLOAT EnvLoEdge, FLOAT EnvHiEdge, FLOAT BandsPerOctave, UINT16 SpectralFlatness, FLOAT FlatLoEdge, FLOAT FlatHiEdge, UINT16 ScaleRatio, UINT16 EvalMeans, UINT16 EvalVariances, RESOURCE OutputResource)

Version: 1.0
Description: Extracts MPEG-7 Low Level Descriptors from an audio file
Parameter List
Name: InputResource
Description: The Resource to be analyzed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-aiff wav aiff basic
Ranges:
Name: HopSize
Description: Time (in seconds) between successive estimation of descriptors
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: AudioPower
Description: Compute AudioPower if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: SpectralCentroid
Description: Compute SpectralCentroid if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: SpectralSpread

Description: Compute SpectralSpread if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:

Name: SpectralEnvelope
Description: Compute SpectralEnvelope if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:

Name: EnvLoEdge
Description: Lower edge of logarithmically-spaced frequency bands
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:

Name: EnvHiEdge
Description: Higher edge of logarithmically-spaced frequency bands
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:

Name: BandsPerOctave
Description: Frequency resolution of logarithmic spectrum [0.125,0.25,0.5,1,2,4,8,16,32]
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:

Name: SpectralFlatness
Description: Compute SpectralFlatness if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:

Name: FlatLoEdge
Description: Lower edge of logarithmically-spaced frequency bands
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:

Name: FlatHiEdge
Description: Higher edge of logarithmically-spaced frequency bands
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:

Name: ScaleRatio

Description: Scaling Ratio for scaling operations (means and variances)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: EvalMeans
Description: Compute means of descriptors if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: EvalVariances
Description: Compute variances of descriptors if set to 1
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputResource
Description: Where the produced MPEG-7 description will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: SUCCESS or ERROR followed by a message in case of error

6.2.2 Segmentation

STRING Segmentation (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Detect silence/noise/speech/music segments
Parameter List
Name: InputResource
Description: The Resource to be analyzed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-aiff wav aiff basic
Ranges:
Name: OutputResource
Description: Where the produced MPEG-7 description will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: SUCCESS or ERROR followed by a message in case of error

6.2.3 TempoEstimation

STRING TempoEstimation (RESOURCE InputResource, FLOAT BpmLoLimit, FLOAT BpmHiLimit, RESOURCE OutputResource)

Version: 1.0
Description: Estimate tempo (in BPM) of a music file
Parameter List
Name: InputResource
Description: The Resource to be analyzed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-aiff wav aiff basic
Ranges:
Name: BpmLoLimit
Description: Minimum acceptable tempo in beats per minute
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: BpmHiLimit
Description: Maximum acceptable tempo in beats per minute
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputResource
Description: Where the produced MPEG-7 description will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: SUCCESS or ERROR followed by a message in case of error

6.2.4 MusicGenreEstimation

STRING MusicGenreEstimation (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Estimate the music genre of a music file
Parameter List
Name: InputResource
Description: The Resource to be analyzed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav x-aiff wav aiff basic
Ranges:
Name: OutputResource
Description: Where the produced MPEG-7 description will be stored
Paramater Type RESOURCE

Default Value:
Constraints:

Result: Result
Result type: STRING
Result Description: SUCCESS or ERROR followed by a message in case of error

6.3 Audio Fingerprint Extraction Plug-in

Category: ContentProcessing
Identifier: AudioFingerprintExtraction
Library: AudioFingerprintExtraction **Version:** 1.001
Vendor: Axmedis
Main Library: audioFPplugin.dll

Description: Audio Fingerprint Extraction

FunctionList:
- AxAFPExtract
- AxAFPCompare

6.3.1 AxAFPExtract

STRING AxAFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 nFeatures, INT32 frameSize, INT32 frameShift, INT32 offset)

Version: 1.0
Description: Extracts a fingerprint of the first audio stream in a given Multimedia File(Audio or Video).
Parameter List
Name: InputResource
Description: The Resource to extract the fingerprint from
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-wav wav x-aiff x-ms-wma mpeg video
Resource Format: x-msvideo mpeg x-ms-wmv quicktime avi
Ranges:
Name: OutputResource
Description: Where the fingerprint bitmap image will be stored (choose new resource)
Paramater Type RESOURCE
Default Value:
Constraints:
Name: OutputResource2
Description: Where the binary fingerprint resource will be stored(choose new resource)
Paramater Type RESOURCE
Default Value:
Constraints:
Name: nFeatures
Description: Number of Features for the finger print. 18 is the desired Standard
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: frameSize
Description: Size of the windowing Size for the Subfingerprints
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: frameShift

Description: Frame overlap for the subfingerprints
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: offset
Description: Frame offset for the fingerprint calculation
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.3.2 AxAFPCompare

STRING AxAFPCompare (RESOURCE InputResource, RESOURCE InputResource2, INT32 CLENGTH, INT32 RLENGTH, FLOAT MINBER, INT32 TIMEPOS)

Version: 1.0
Description: Compares two audio fingerprint files and returns the lowest BER and time position.
Parameter List
Name: InputResource
Description: The candidate object to compare
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: fingerprint
Resource Format: audio
Ranges:
Name: InputResource2
Description: The reference object to compare
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: fingerprint
Resource Format: audio
Ranges:
Name: CLENGTH
Description: Length of the candidate Object(in seconds)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: RLENGTH
Description: Length of the reference Object(in seconds)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:

Name: MINBER
Description: Minimal BER found (in percent)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: TIMEPOS
Description: Time position for the minimal BER found (in seconds)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result the comparison, SUCCESS if ok, ERROR followed by a message in case of error

6.4 Text Descriptors Plug-in

Category: ContentProcessing
Identifier: TextDescriptors
Library: Descriptor **Version:** 1.001
Vendor: Axmedis
Main Library: descriptorextractorplugin.dll

Description: Plugin for descriptors extraction from text documents

FunctionList:
- KWFromComparisons
- KWFromSemanticAnalysis

6.4.1 KWFromComparisons

STRING KWFromComparisons (RESOURCE InputResource, UINT16 MaxKWNumber, BOOLEAN DetailedResults, STRING Keywords)

Version: 1.0
Description: Retrieves the keywords exploiting frequency lists corpus only.
Parameter List
Name: InputResource
Description: The Resource to be processed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:
Name: MaxKWNumber
Description: How many keyword requested as a maximum.
Paramater Type UINT16
Default Value:
Constraints:
Name: DetailedResults
Description: Returns keywords with ranking values
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: Keywords
Description: A string containing keywords separated by carriage return + newline chars
Paramater Type STRING
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.4.2 KWFromSemanticAnalysis

STRING KWFromSemanticAnalysis (RESOURCE InputResource, UINT16 MaxKWNumber, STRING Keywords)

Version: 1.0
Description: Retrieves the keywords exploiting frequency lists corpus and WordNet synsets

relations.

Parameter List

Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:

Name: MaxKWNumber
Description: How many keyword requested as a maximum.
Paramater Type UINT16
Default Value:
Constraints:

Name: Keywords
Description: A string containing keywords separated by carriage return + newline chars'
Paramater Type STRING
Default Value:
Constraints:

Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.5 Text to Docs Adaptation Plug-in

Category: ContentProcessing
Identifier: TextDocsAdaptation
Library: Adaptation **Version:** 1.001
Vendor: Axmedis
Main Library: documentadaptation.dll

Description: Plugin for transcoding text documents

FunctionList:
- DocumentConversion

6.5.1 DocumentConversion

STRING DocumentConversion (RESOURCE InputResource, STRING ConversionFormat, RESOURCE OutputResource)

Version: 1.0
Description: Transcodes the given text document in the supplied format.
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: application
Resource Format: pdf text
Resource Format: html application
Resource Format: postscript application
Resource Format: rtf application
Resource Format: msword text
Resource Format: plain
Ranges:
Name: ConversionFormat
Description: The format to which the resource will be converted.
Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.6 TextFingerprint

Category: ContentProcessing
Identifier: TextFingerprint
Library: Fingerprint **Version:** 0.2.0
Vendor: Axmedis
Main Library: text-fingerprint.dll

Description: Plugin for text fingerprinting

FunctionList:

- MD5Hash

6.6.1 MD5Hash

STRING MD5Hash (RESOURCE Source)

Version: 1.0
Description: Generates md5 hash from a textual document
Parameter List
Name: Source
Description: The source document
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:
Result: Result
Result type: STRING
Result Description: md5 hash

6.7 Multimedia Adaptation plug-in

Category: ContentProcessing
Identifier: MultimediaAdaptation
Library: MultimediaAdaptation Version: 1.001
Vendor: Axmedis
Main Library: multimediaadaptationplugin.dll

Description: Plugin for multimedia processing

FunctionList:
- Mp4To3gp
- Mp4ToISMA
- AddMultimediaFiles
- ToMp4
- ExtractMediaTrack
- CatMultimediaFiles
- DelayTrack
- RemoveTrack
- ExtractFromStartToEnd
- Mp4ToAvi

6.7.1 Mp4To3gp

STRING Mp4To3gp (RESOURCE InputResource, RESOURCE OutputResource, BOOLEAN KeepSys)

Version: 1.0
Description: Convert an mp4 file to 3gp
Parameter List
Name: InputResource
Description: Input File to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Name: KeepSys
Description: Keep systems tracks
Paramater Type BOOLEAN
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of 3gp conversion, SUCCESS if ok, ERROR followed by a

message in case of error

6.7.2 Mp4ToISMA

STRING Mp4ToISMA (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Convert an mp4 file to ISMA specification
Parameter List
Name: InputResource
Description: File to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of ISMA conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.7.3 AddMultimediaFiles

STRING AddMultimediaFiles (RESOURCE InputResource, RESOURCE BaseResource, UINT32 Delay, DOUBLE ImportLength, STRING TrackID, UINT32 FPS, STRING Lang, RESOURCE OutputResource)

Version: 1.0
Description: Import a media file into a multimedia file (samples are added to new tracks)
Parameter List
Name: InputResource
Description: The media to add
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg x-aac amr evrc smv x-vorbis x-gsm video
Resource Format: x-media x-nhnt x-info x-cmp x-m4v x-h263 x-h264 x-msvideo mpeg dvd mp4 mp4v-es 3gpp xmt xmta mp4-bt vrml image
Resource Format: jpeg png application
Resource Format: x-cdlink text
Resource Format: srt sub ttxt texml model
Resource Format: x3d+xml x3d+vrml
Ranges:
Name: BaseResource
Description: Base resource where to add the new media

Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg x-aac amr evrc smv x-vorbis x-gsm video
Resource Format: x-media x-nhnt x-info x-cmp x-m4v x-h263 x-h264 x-msvideo mpeg dvd mp4 mp4v-es 3gpp xmt xmta mp4-bt vrml image
Resource Format: jpeg png application
Resource Format: x-cdlink text
Resource Format: srt sub ttxt texml model
Resource Format: x3d+xml x3d+vrml
Ranges:

Name: Delay
Description: Delay in milliseconds of the new track
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:

Name: ImportLength
Description: Number of seconds to import from input file (starting from the beginning)
Paramater Type DOUBLE
Default Value:
Constraints:
Resource Type:
Ranges:

Name: TrackID
Description: Track to extract
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:

Name: FPS
Description: Frames per sample
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:

Name: Lang
Description: Language code
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:

Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:

Result: Result
Result type: STRING

Result Description: The result of adding files, SUCCESS if ok, ERROR followed by a message in case of error

6.7.4 ToMp4

STRING ToMp4 (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Import a media file into a multimedia file (samples are added to new tracks)
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg x-aac amr evrc smv x-vorbis x-gsm video
Resource Format: x-media x-nhnt x-info x-cmp x-m4v x-h263 x-h264 x-msvideo mpeg dvd mp4 mp4v-es 3gpp xmt xmta mp4-bt vrml image
Resource Format: jpeg png application
Resource Format: x-cdlink text
Resource Format: srt sub ttxt texml model
Resource Format: x3d+xml x3d+vrml
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of mp4 conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.7.5 ExtractMediaTrack

STRING ExtractMediaTrack (RESOURCE InputResource, RESOURCE OutputResource, UINT32 TrackID, STRING Mimetype)

Version: 1.0
Description: Extracts a track into a new file
Parameter List
Name: InputResource
Description: Input file to extract track
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es 3gpp
Ranges:
Name: OutputResource
Description: Where the produced resource (track) will be stored
Paramater Type RESOURCE
Default Value:

Constraints:
Resource Type:
Ranges:
Name: TrackID
Description: Track to extract
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: Mimetype
Description: Mimetype for output resource
Paramater Type STRING
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of the extraction, SUCCESS if ok, ERROR followed by a message in case of error

6.7.6 CatMultimediaFiles

STRING CatMultimediaFiles (RESOURCE InputResourceA, RESOURCE InputResourceB, RESOURCE OutputResource)

Version: 1.0
Description: Concatenates two multimedia files
Parameter List
Name: InputResourceA
Description: Input file A to concatenate
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg x-aac amr evrc smv x-vorbis x-gsm video
Resource Format: x-media x-nhnt x-info x-cmp x-m4v x-h263 x-h264 x-msvideo mpeg dvd mp4 mp4v-es 3gpp xmt xmta mp4-bt vrml image
Resource Format: jpeg png application
Resource Format: x-cdlink text
Resource Format: srt sub ttxt texml model
Resource Format: x3d+xml x3d+vrml
Ranges:
Name: InputResourceB
Description: Input file B to concatenate
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg x-aac amr evrc smv x-vorbis x-gsm video
Resource Format: x-media x-nhnt x-info x-cmp x-m4v x-h263 x-h264 x-msvideo mpeg dvd mp4 mp4v-es 3gpp xmt xmta mp4-bt vrml image
Resource Format: jpeg png application
Resource Format: x-cdlink text
Resource Format: srt sub ttxt texml model
Resource Format: x3d+xml x3d+vrml
Ranges:

Name: OutputResource
Description: Result file
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of the concatenation, SUCCESS if ok, ERROR followed by a message in case of error

6.7.7 DelayTrack

STRING DelayTrack (RESOURCE InputResource, UINT32 Delay, STRING TrackID, RESOURCE OutputResource)

Version: 1.0
Description: Delay a track
Parameter List
Name: InputResource
Description: File to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es 3gpp
Ranges:
Name: Delay
Description: Delay in milliseconds of the track
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: TrackID
Description: Track to delay
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of track delay, SUCCESS if ok, ERROR followed by a message in case of error

6.7.8 RemoveTrack

STRING RemoveTrack (RESOURCE InputResource, STRING TrackID, RESOURCE OutputResource)

Version: 1.0
Description: Remove a track
Parameter List
Name: InputResource
Description: File to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es 3gpp
Ranges:
Name: TrackID
Description: Track to remove
Paramater Type STRING
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of track removing, SUCCESS if ok, ERROR followed by a message in case of error

6.7.9 ExtractFromStartToEnd

STRING ExtractFromStartToEnd (RESOURCE InputResource, DOUBLE Start, DOUBLE End, RESOURCE OutputResource)

Version: 1.0
Description: Extracts a file from start to end
Parameter List
Name: InputResource
Description: File to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: mp4 video
Resource Format: mp4 mp4v-es 3gpp
Ranges:
Name: Start
Description: Start of extraction in seconds
Paramater Type DOUBLE
Default Value:
Constraints:
Resource Type:

Ranges:
Name: End
Description: End of extraction in seconds
Paramater Type DOUBLE
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of the extraction by time, SUCCESS if ok, ERROR followed by a message in case of error

6.7.10 Mp4ToAvi

STRING Mp4ToAvi (RESOURCE InputResource, FLOAT FPS, UINT32 Width, UINT32 Height, RESOURCE OutputResource)

Version: 1.0
Description: Convert a MP4 BIFS pure file to AVI
Parameter List
Name: InputResource
Description: Input MP4 BIFS pure file(no audio, no image, no video) to convert to AVI
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: video
Resource Format: mp4 mp4v-es
Ranges:
Name: FPS
Description: Extraction framerate (default:0 computed from the BIFS track duration)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: Width
Description: Width of the bifs scene (default:0 original size)
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: Height
Description: Height of the bifs scene (default:0 original size)
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:

Ranges:
Name: OutputResource
Description: Result file
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of the conversion to avi, SUCCESS if ok, ERROR followed by a message in case of error

6.8 Ringtone Adaptation plug-in

Category: ContentProcessing
Identifier: RingtoneAdaptation
Library: RingtoneAdaptation **Version:** 1.001
Vendor: Axmedis
Main Library: ringtoneadaptationplugin.dll

Description: Plugin for Ring Tone processing

FunctionList:
- convert
- convert to MP3
- convert to WAV
- resample
- convert and resample
- getInfo
- clip

6.8.1 convert

STRING convert (RESOURCE InputResource, STRING Mimetype, RESOURCE OutputResource)

Version: 1.0
Description: Convert from one format to another
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: Mimetype
Description: Mimetype for output resource
Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.8.2 convert_to_MP3

STRING convert_to_MP3 (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Convert from one of the supported formats to MP3 Format
Parameter List

Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.8.3 convert_to_WAV

STRING convert_to_WAV (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Convert from one of the supported formats to WAV Format
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.8.4 resample

STRING resample (RESOURCE InputResource, RESOURCE OutputResource, UINT32 OutputSamplingRate, UINT16 OutputNumChannels, UINT16 OutputBitRate)

Version: 1.0
Description: Resample the input file (ie changing frequecy, bitrate etc). Please note that some of the bit rates and channels won't go together and that will result in an exception. In that case you have to restart the plugin.
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:

Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: OutputSamplingRate
Description: Sampling rate of the output audio file (default: sampling rate of the input)
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputNumChannels
Description: Number of channels of the output audio file (default: number of channels of the input)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputBitRate
Description: Bit rate of the output audio file - Only applies to compressed audio file formats (default: 64 kb)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.8.5 convert and resample

STRING convert_and_resample (RESOURCE InputResource, STRING Mimetype, RESOURCE OutputResource, UINT32 OutputSamplingRate, UINT16 OutputNumChannels, UINT16 OutputBitRate)

Version: 1.0
Description: Convert and Resample the input file (ie converting to any supported format and changing frequecy, bitrate etc, all at the same time. Please note that some of the bit rates and channels won't go together and that will result in an exception. In that case you have to restart the plugin.
Parameter List
Name: InputResource
Description: The Resource to be converted and resampled
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: Mimetype
Description: Mimetype for output resource

Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: OutputSamplingRate
Description: Sampling rate of the output audio file (default: sampling rate of the input)
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputNumChannels
Description: Number of channels of the output audio file (default: number of channels of the input)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: OutputBitRate
Description: Bit rate of the output audio file - Only applies to compressed audio file formats (default: 64 kb)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.8.6 getInfo

STRING getInfo (RESOURCE InputResource, STRING Mimetype, UINT32 SamplingRate, UINT16 NumChannels, UINT16 BitRate, STRING Duration)

Version: 1.0
Description: Get all the information about the input Ring Tone
Parameter List
Name: InputResource
Description: The Resource whose Information is to be retrieved
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio mid
Ranges:
Name: Mimetype
Description: Mimetype for input resource
Paramater Type STRING
Default Value:
Constraints:

Resource Type:
Ranges:
Name: SamplingRate
Description: Sampling rate of the input ring tone
Paramater Type UINT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: NumChannels
Description: Number of channels of the input ring tone
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: BitRate
Description: Bit rate of the input ring tone - (default: 64 kb)
Paramater Type UINT16
Default Value:
Constraints:
Resource Type:
Ranges:
Name: Duration
Description: Duration of the Ringtone (hrs:mins:secs: msecs)
Paramater Type STRING
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of the operation, SUCCESS if ok, ERROR followed by a message in case of error

6.8.7 clip

STRING clip (RESOURCE InputResource, RESOURCE OutputResource, STRING Mimetype, FLOAT ReadStartingTime, FLOAT ReadEndingTime)

Version: 1.0
Description: Clip the file for the specified time (for e.g. reducing it to a 30 sec clip)
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: audio
Resource Format: basic x-wav x-aiff x-ms-wma x-mpeg x-vorbis x-pn-realaudio
Ranges:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: Mimetype

Description: Mimetype for output resource
Paramater Type STRING
Default Value:
Constraints:
Name: ReadStartingTime
Description: Starting time for the clip(default: beginning of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: ReadEndingTime
Description: Ending time for the clip (default: end of the file)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.9 Audio recognition and monitoring Plugin

To be added in the next version

6.10 Image Processing Plugin

This plug in uses the ImageMagick Library, please for additional information refer to user manual of that library and tools.

6.10.1 Main functionalities

The image processing plug-in allows adapting image resources to various use case. For example it can be used to convert different image formats, to apply various effects, to resize, to mirror, etc. In total the plug-in is composed of forty-one functions that are:

- **Conversion**, to convert the image
- **Import**, to import an image
- **Resize**, to resize the image
- **Contrast**, to change the image contrast
- **Edge**, to highlight edges of the image
- **Embross**, to highlight edges with 3D effect
- **Blur**, to blur the image
- **GaussianBlur**, to apply a Gaussian Blur to the image
- **Median**, to apply a median filter to the image
- **Mirror**, to mirror the image
- **Noise**, to add noise in the image
- **Despeckle**, to reduce the noise from the image using the despeckle filter
- **Equalize**, to apply an histogram equalization to the image
- **Enhance**, to minimize the noise of the image
- **ExtractChannel**,
- **GrayScale**, to convert a coloured image to grayscale
- **Magnify**, to scale up the image
- **Minify**, to scale down the image
- **Modulate**, to modulate hue, saturation, and brightness of the image
- **Monochrome**, to create a monochrome image
- **Negate**, to negate colours in the image
- **Normalize**, to increase contrast by normalizing the pixel values
- **OilPaint**, to create a image looks like oil painting
- **Quality**, to change the JPEG/MIFF/PNG compression
- **Quantize**, to set the preferred number of colours in the image
- **Raise**, to highlight or dark the edges of an image to give a 3D raised or lowered effect
- **ReduceNoise**, to reduce the noise of the image
- **Replace**, to replace the image
- **FloodFill**, to apply a flood-fill texture
- **Rool**, to roll the image by a specified number of columns and rows
- **Rotate**, to rotate the image specifying a number of degrees
- **Scale**, to scale the image by using a specified ratio
- **Shear**, to create a parallelogram by sliding the image by X of Y axis
- **Shade**, to shade the image using distant light source
- **Spread**, to spread pixels randomly
- **SetOpacity** to set the opacity of the image
- **SubImage**,

- **GetInfo**, to see information about the image
- **SetMaskColour,**
- **Paste,** to paste the image
- **Test,** to test the image

A More detailed description of these functionalities is available in section **Errore. L'origine riferimento non è stata trovata.**

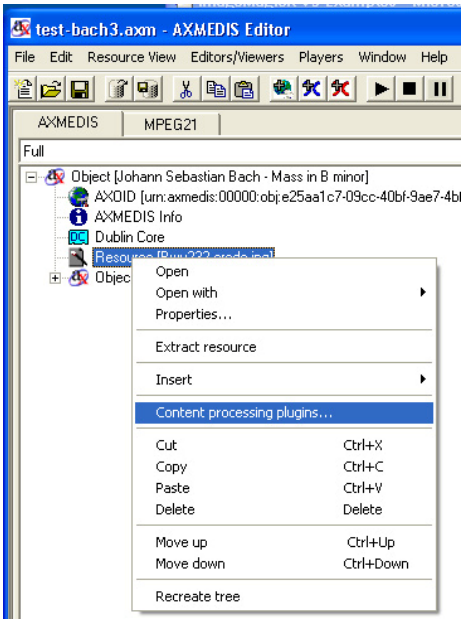
6.10.2 Relationship with other tools

This tool is implemented as a plug-in. Like other plug-ins, its functionality is available via the AXMEDIS Editor and the AXMEDIS Processing Engine.

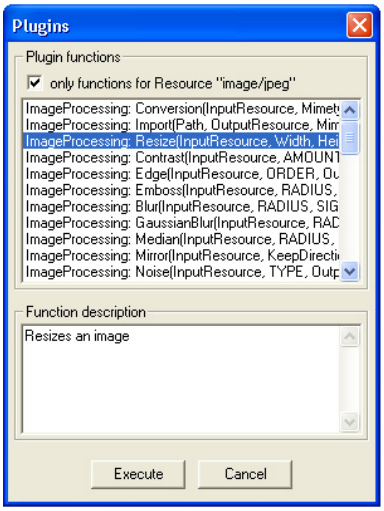
6.10.3 Detailed description of the functionalities and Screenshots

Here's an example on how to use the plug-in with AXMEDIS Editor.

The plug-in can be applied to all images resources in all formats embedded into an AXMEDIS object. Selecting one resource in the tree and right clicking, select **Content Processing plugins…**



A new dialog will appear will the list of available functionalities. Selecting a functionality will a appear a brief description in the **Function description** box.



Selecting the appropriate function and pressing the **Execute** button a new dialog appears with a number of fields to be filled-in. the aspect of the dialog and the number of fields is different for each function.

Please, refer to section ?? for a detailed description of the values needed for each functionalities.

In the **Output Resource** cascading menu is possible to decide if the function will produce a new resource or will overwrite the old one.

Here's a brief analysis of image processing functionalities.

Since the image processing plug-in is based on the GPL source code of ImageMagick, for a more detailed description of these functionalities, please refer to the following links:

- ImageMagick website: http://www.imagemagick.org/script/index.php
- **The Definitive Guide to ImageMagick** by Michael Still, available on: http://www.amazon.com/Definitive-Guide-ImageMagick/dp/1590595904/sr=8-1/qid=1157030444/ref=pd_bbs_1/104-0533291-5821542?ie=UTF8
- Examples of ImageMagick usage are available here: http://www.cit.gu.edu.au/~anthony/graphics/imagick6

6.10.3.1 Conversion

STRING Conversion (RESOURCE InputResource, STRING Mimetype, RESOURCE OutputResource)

Version: 1.0
Description: Convert an image
Parameter List
Name: InputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: Mimetype
Description: Mimetype for output resource
Paramater Type STRING
Default Value:

Constraints:
Name: OutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.2 Import

STRING Import (STRING Path, RESOURCE OutputResource, STRING MimeType)

Version: 1.0
Description: Import an image
Parameter List
Name: Path
Description: Path to the image
Paramater Type STRING
Default Value:
Constraints:
Name: OutputResource
Description: Where the imported resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: MimeType
Description: Test
Paramater Type STRING
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.3 Resize

STRING Resize (RESOURCE InputResource, INT32 Width, INT32 Height, BOOLEAN KeepAspectRatio, RESOURCE OutputResource)

Version: 1.0
Description: Resizes an image
Parameter List
Name: InputResource
Description: The Resource to be resized
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image

Resource Format: jpeg gif png
Ranges:
Name: Width
Description: The new image width
Paramater Type INT32
Default Value:
Constraints:
Name: Height
Description: The new image height
Paramater Type INT32
Default Value:
Constraints:
Name: KeepAspectRatio
Description: Indicates to preserve image aspect ratio or not
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: OutputResource
Description: Where the resized resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.4 Contrast

STRING Contrast (RESOURCE InputResource, INT32 AMOUNT, RESOURCE OutputResource)

Version: 1.0
Description: Change image contrast
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: AMOUNT
Description: The contrast amount
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in

case of error

6.10.3.5 Edge

STRING Edge (RESOURCE InputResource, INT32 ORDER, RESOURCE OutputResource)

Version: 1.0
Description: Edge image (highlight edges in image). The radius is the radius of the pixel neighbourhood.. Specify a radius of zero for automatic radius selection.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: ORDER
Description: The Order Edge
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.6 Emboss

STRING Emboss (RESOURCE InputResource, INT32 RADIUS, INT32 SIGMA, RESOURCE OutputResource)

Version: 1.0
Description: Emboss image (highlight edges with 3D effect). The radius_ parameter specifies the radius of the Gaussian, in pixels, not counting the center pixel. The sigma_ parameter specifies the standard deviation of the Laplacian, in pixels.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: RADIUS
Description: The Radius Emboss
Paramater Type INT32
Default Value:

Constraints:
Name: SIGMA
Description: The sigma Emboss
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.7 Blur

STRING Blur (RESOURCE InputResource, INT32 RADIUS, INT32 SIGMA, RESOURCE OutputResource)

Version: 1.0
Description: Blur image. The radius_ parameter specifies the radius of the Gaussian, in pixels, not counting the center pixel. The sigma_ parameter specifies the standard deviation of the Laplacian, in pixels.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: RADIUS
Description: The Radius Blur
Paramater Type INT32
Default Value:
Constraints:
Name: SIGMA
Description: The sigma Blur
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.8 GaussianBlur

STRING GaussianBlur (RESOURCE InputResource, INT32 RADIUS, INT32 SIGMA, RESOURCE OutputResource)

Version: 1.0
Description: GaussianBlur the image
Parameter List
Name: InputResource
Description: Gaussian blur image. The number of neighbor pixels to be included in the convolution mask is specified by 'width_'. For example, a width of one gives a (standard) 3x3 convolution mask. The standard deviation of the gaussian bell curve is specified by 'sigma'.
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: RADIUS
Description: The Radius GaussianBlur
Paramater Type INT32
Default Value:
Constraints:
Name: SIGMA
Description: The sigma GaussianBlur
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.9 Median

STRING Median (RESOURCE InputResource, INT32 RADIUS, RESOURCE OutputResource)

Version: 1.0
Description: Median the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: RADIUS
Description: The Radius Median
Paramater Type INT32
Default Value:

Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.10 Mirror

STRING Mirror (RESOURCE InputResource, BOOLEAN KeepDirection, RESOURCE OutputResource)

Version: 1.0
Description: Mirror the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: KeepDirection
Description: The KeepDirection Mirror
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.11 Noise

STRING Noise (RESOURCE InputResource, INT32 TYPE, RESOURCE OutputResource)

Version: 1.0
Description: Noise the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image

Resource Format: jpeg gif png
Ranges:
Name: TYPE
Description: The Type Noise
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.12 Despeckle

STRING Despeckle (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Despeckle image (reduce speckle noise)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.13 Equalize

STRING Equalize (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Equalize image (histogram equalization)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image

Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.14 Enhance

STRING Enhance (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Enhance image (minimize noise)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.15 ExtractChannel

STRING ExtractChannel (RESOURCE InputResource, INT32 CHANNEL, RESOURCE OutputResource)

Version: 1.0
Description: ExtractChannel the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: CHANNEL
Description: The Channel ExtractChannel
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in

case of error

6.10.3.16 Grayscale

STRING Grayscale (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Grayscale the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.17 Magnify

STRING Magnify (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Magnify image by integral size
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.18 Minify

STRING Minify (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Reduce image by integral size
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.19 Modulate

STRING Modulate (RESOURCE InputResource, INT32 BRIGHTNESS, INT32 SATURATION, INT32 HUE, RESOURCE OutputResource)

Version: 1.0
Description: Modulate percent hue, saturation, and brightness of an image.Modulation of saturation and brightness is as a ratio of the current value(1.0 for no change). Modulation of hue is an absolute rotation of -180 degrees to +180 degrees from the current position corresponding to an argument range of 0 to 2.0 (1.0 for no change).
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: BRIGHTNESS
Description: Brightness modulate
Paramater Type INT32
Default Value:
Constraints:
Name: SATURATION
Description: Saturation modulate
Paramater Type INT32
Default Value:
Constraints:
Name: HUE

Description: Hue modulate
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.20 Monochrome

STRING Monochrome (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Monochrome the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.21 Negate

STRING Negate (RESOURCE InputResource, BOOLEAN GRAYSCALE, RESOURCE OutputResource)

Version: 1.0
Description: Negate colors in image. Replace every pixel with its complementary color (white becomes black, yellow becomes blue, etc.). Set grayscale to only negate grayscale values in image.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png

Ranges:
Name: GRAYSCALE
Description: Where the manipulated resource will be stored
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.22 Normalize

STRING Normalize (RESOURCE InputResource, RESOURCE OutputResource)

Version: 1.0
Description: Normalize image (increase contrast by normalizing the pixel values to span the full range of color values)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.23 OilPaint

STRING OilPaint (RESOURCE InputResource, INT32 RADIUS, RESOURCE OutputResource)

Version: 1.0
Description: Oilpaint image (image looks like oil painting)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image

Resource Format: jpeg gif png
Ranges:
Name: RADIUS
Description: the radius OilPaint
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.24 Quality

STRING Quality (RESOURCE InputResource, INT32 LEVEL, RESOURCE OutputResource)

Version: 1.0
Description: JPEG/MIFF/PNG compression level (default 75).
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: LEVEL
Description: the quality of the compress level
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.25 Quantize

STRING Quantize (RESOURCE InputResource, INT32 NCOLORS, RESOURCE OutputResource)

Version: 1.0
Description: Preferred number of colors in the image. The actual number of colors in the image may be less than your request, but never more. Images with less unique colors than specified with this option will have any duplicate or unused colors removed.

Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: NCOLORS
Description: the number of color
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.26 Raise

STRING Raise (RESOURCE InputResource, INT32 WIDTH, INT32 HEIGHT, INT32 XOFFSET, INT32 YOFFSET, BOOLEAN RISED, RESOURCE OutputResource)

Version: 1.0
Description: Raise image (lighten or darken the edges of an image to give a 3-D raised or lowered effect)
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: WIDTH
Description: The width is parts of the geometry specification are measured in pixels
Paramater Type INT32
Default Value:
Constraints:
Name: HEIGHT
Description: The height is parts of the geometry specification are measured in pixels
Paramater Type INT32
Default Value:
Constraints:
Name: XOFFSET
Description: The left edge of the object is to be placed xoffset pixels in from the left edge of the image.
Paramater Type INT32

Default Value:
Constraints:
Name: YOFFSET
Description: The top edge of the object is to be yoffset pixels below the top edge of the image.
Paramater Type INT32
Default Value:
Constraints:
Name: RISED
Description: raisedFlag
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.27 ReduceNoise

STRING ReduceNoise (RESOURCE InputResource, INT32 ORDER, RESOURCE OutputResource)

Version: 1.0
Description: Reduce noise in image using a noise peak elimination filter.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: ORDER
Description: order
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.28 Replace

STRING Replace (RESOURCE InputResource, INT32 R1, INT32 G1, INT32 B1, INT32 R2, INT32

G2, INT32 B2, RESOURCE OutputResource)

Version: 1.0
Description: Replace the image
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: R1
Description: r1
Paramater Type INT32
Default Value:
Constraints:
Name: G1
Description: g1
Paramater Type INT32
Default Value:
Constraints:
Name: B1
Description: b1
Paramater Type INT32
Default Value:
Constraints:
Name: R2
Description: r2
Paramater Type INT32
Default Value:
Constraints:
Name: G2
Description: g2
Paramater Type INT32
Default Value:
Constraints:
Name: B2
Description: b2
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.29 FloodFill

STRING FloodFill (RESOURCE InputResource, INT32 X, INT32 Y, INT32 B, INT32 R, INT32 G, RESOURCE OutputResource)

Version: 1.0
Description: Flood-fill texture across pixels that match the color of the target pixel and are neighbors of the target pixel. Uses current fuzz setting when determining color match.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: X
Description: x
Paramater Type INT32
Default Value:
Constraints:
Name: Y
Description: y
Paramater Type INT32
Default Value:
Constraints:
Name: B
Description: b
Paramater Type INT32
Default Value:
Constraints:
Name: R
Description: r
Paramater Type INT32
Default Value:
Constraints:
Name: G
Description: g
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.30 Roll

STRING Roll (RESOURCE InputResource, INT32 X, INT32 Y, RESOURCE OutputResource)

Version: 1.0

Description: Roll image (rolls image vertically and horizontally) by specified number of columnms and rows)

Parameter List

Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:

Name: X
Description: x
Paramater Type INT32
Default Value:
Constraints:

Name: Y
Description: y
Paramater Type INT32
Default Value:
Constraints:

Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:

Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.31 Rotate

STRING Rotate (RESOURCE InputResource, INT32 ANGLE, RESOURCE OutputResource)

Version: 1.0
Description: Rotate image counter-clockwise by specified number of degrees.
Parameter List

Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:

Name: ANGLE
Description: Number of the degrees
Paramater Type INT32
Default Value:
Constraints:

Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:

Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.32 Scale

STRING Scale (RESOURCE InputResource, INT32 WIDTH, INT32 HEIGHT, INT32 MODE, RESOURCE OutputResource)

Version: 1.0
Description: Resize image by using simple ratio algorithm
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: WIDTH
Description: Width
Paramater Type INT32
Default Value:
Constraints:
Name: HEIGHT
Description: Height
Paramater Type INT32
Default Value:
Constraints:
Name: MODE
Description: Mode
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.33 Shear

STRING Shear (RESOURCE InputResource, INT32 XSHEAR, INT32 Yshear, RESOURCE OutputResource)

Version: 1.0
Description: Shear image (create parallelogram by sliding image by X or Y axis). Shearing slides one edge of an image along the X or Y axis, creating a parallelogram. An X direction shear slides an edge along

the X axis, while a Y direction shear slides an edge along the Y axis. The amount of the shear is controlled by a shear angle. For X direction shears, x degrees is measured relative to the Y axis, and similarly, for Y direction shears y degrees is measured relative to the X axis. Empty triangles left over from shearing the image are filled with the color defined as borderColor.

Parameter List

Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:

Name: XSHEAR
Description: XSHEAR
Paramater Type INT32
Default Value:
Constraints:

Name: Yshear
Description: Yshear
Paramater Type INT32
Default Value:
Constraints:

Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:

Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.34 Shade

STRING Shade (RESOURCE InputResource, INT32 AZIMUTH, INT32 ELEVATION, BOOLEAN COLOR, RESOURCE OutputResource)

Version: 1.0

Description: Shade image using distant light source. Specify azimuth_ and elevation_ as the position of the light source. By default, the shading results as a grayscale image.. Set colorShading_ to true to shade the red, green, and blue components of the image.

Parameter List

Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:

Name: AZIMUTH
Description: AZIMUTH
Paramater Type INT32
Default Value:

Constraints:
Name: ELEVATION
Description: ELEVATION
Paramater Type INT32
Default Value:
Constraints:
Name: COLOR
Description: COLOR
Paramater Type BOOLEAN
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.35 Spread

STRING Spread (RESOURCE InputResource, INT32 AMOUNT, RESOURCE OutputResource)

Version: 1.0
Description: Spread pixels randomly within image by specified amount.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: AMOUNT
Description: AMOUNT
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.36 SetOpacity

STRING SetOpacity (RESOURCE InputResource, INT32 LEVEL, RESOURCE OutputResource)

Version: 1.0
Description: Set the opacity of the image.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: LEVEL
Description: LEVEL
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.37 SubImage

STRING SubImage (RESOURCE InputResource, INT32 X, INT32 Y, INT32 WIDTH, INT32 HEIGHT, RESOURCE OutputResource)

Version: 1.0
Description: SubImage image.
Parameter List
Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: X
Description: x coordinate of the top-level corner of the rectangle
Paramater Type INT32
Default Value:
Constraints:
Name: Y
Description: y coordinate of the top-level corner of the rectangle
Paramater Type INT32
Default Value:
Constraints:
Name: WIDTH
Description: Width member
Paramater Type INT32

Default Value:
Constraints:
Name: HEIGHT
Description: Height member
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.38 GetInfo

STRING GetInfo (RESOURCE InputResource, INT32 WIDTH, INT32 HEIGHT)

Version: 1.0
Description: Return the size of the image.
Parameter List
Name: InputResource
Description: The Resource under analisys
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: WIDTH
Description: The width of the Image
Paramater Type INT32
Default Value:
Constraints:
Name: HEIGHT
Description: The height of the Image
Paramater Type INT32
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.39 SetMaskColour

STRING SetMaskColour (RESOURCE InputResource, INT32 R, INT32 G, INT32 B, RESOURCE OutputResource)

Version: 1.0
Description: Set the color
Parameter List

Name: InputResource
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: R
Description: Red
Paramater Type INT32
Default Value:
Constraints:
Name: G
Description: Green
Paramater Type INT32
Default Value:
Constraints:
Name: B
Description: Blue
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.40 Paste

STRING Paste (RESOURCE InputResource1, RESOURCE InputResource2, INT32 X, INT32 Y, INT32 COMPOSE, RESOURCE OutputResource)

Version: 1.0
Description: Paste image
Parameter List
Name: InputResource1
Description: The Resource to be manipulated
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image
Resource Format: jpeg gif png
Ranges:
Name: InputResource2
Description: The Resource paste
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: image

Resource Format: jpeg gif png
Ranges:
Name: X
Description: X
Paramater Type INT32
Default Value:
Constraints:
Name: Y
Description: Y
Paramater Type INT32
Default Value:
Constraints:
Name: COMPOSE
Description: Compose
Paramater Type INT32
Default Value:
Constraints:
Name: OutputResource
Description: Where the manipulated resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.10.3.41 Test

RESOURCE Test (RESOURCE InputResource, AXOM Axom)
Version: 1.0
Description: Test an image
Parameter List
Name: InputResource
Description: The Resource to be tested
Paramater Type RESOURCE
Default Value:
Constraints:
Name: Axom
Description: The object
Paramater Type AXOM
Default Value:
Constraints:
Result: Result
Result type: RESOURCE
Result Description: The result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

6.11 VideoAdaptation

Category: ContentProcessing
Identifier: VideoAdaptation
Library: ffmpegTranscoder Version: 1.001
Vendor: Axmedis
Main Library: ffmpegTranscoder.dll

Description: Plugin for video transcoding

FunctionList:

- AX_ffmpegTranscoder

6.11.1 AX_ffmpegTranscoder

STRING AX_ffmpegTranscoder (RESOURCE ExtInputResource, RESOURCE ExtOutputResource, STRING outExt, STRING vidAspectRatio, INT32 vidBitrate, INT32 vidFRate, INT32 vidXFSize, INT32 vidYFSize, INT32 audioBitrate, INT32 audioSamplingrate, INT32 audioChannels, BOOLEAN disableVid, BOOLEAN disableAudio, BOOLEAN sameQ)

Version: 1.0
Description: Adaptation of video files
Parameter List
Name: ExtInputResource
Description: The Resource to be converted
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: video
Resource Format: mpeg avi x-msvideo
Ranges:
Name: ExtOutputResource
Description: Where the produced resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: outExt
Description: Mimetype for output resource(Default: use same mime type as input)
Paramater Type STRING
Default Value:
Constraints:
Name: vidAspectRatio
Description: set aspect ratio (4:3, 16:9 or 1.3333, 1.7777)(Default: same as input)
Paramater Type STRING
Default Value:
Constraints:
Name: vidBitrate
Description: set video bitrate (in kbit/s) (default: sampling rate of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: vidFRate
Description: set frame rate (Hz value)(default: Frame rate of the input)
Paramater Type INT32

Default Value:
Constraints:
Resource Type:
Ranges:
Name: vidXFSize
Description: Width of the frame in pixels (default: width of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: vidYFSize
Description: Heigth of the frame in pixels (default: height of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: audioBitrate
Description: set audio bitrate (in kbit/s)(default: bitrate of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: audioSamplingrate
Description: set audio sampling rate (in Hz) (default: sampling rate of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: audioChannels
Description: set number of audio channels(default: Nr. of channels of the input)
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: disableVid
Description: disable video
Paramater Type BOOLEAN
Default Value:
Constraints:
Resource Type:
Ranges:
Name: disableAudio
Description: disable audio
Paramater Type BOOLEAN
Default Value:
Constraints:
Resource Type:
Ranges:
Name: sameQ
Description: use same video quality as source (implies VBR)
Paramater Type BOOLEAN

Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.12 VideoFingerprintExtraction

Category: ContentProcessing
Identifier: VideoFingerprintExtraction
Library: VideoFingerprintExtraction **Version:** 1.001
Vendor: Axmedis
Main Library: VideoFPPlugIn.dll

Description: Video Fingerprint Extraction

FunctionList:

- AxVFPExtract
- AxVFPCompare

6.12.1 AxVFPExtract

STRING AxVFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 frames)

Version: 1.0
Description: Extracts a fingerprint of a Video stream in a given Multimedia File.
Parameter List
Name: InputResource
Description: The Resource to extract the fingerprint from
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: video
Resource Format: x-msvideo mpeg x-ms-wmv quicktime avi
Ranges:
Name: OutputResource
Description: Where the produced bitmap resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: OutputResource2
Description: Where the produced binary fingerprint resource will be stored
Paramater Type RESOURCE
Default Value:
Constraints:
Name: frames
Description: Number of frames to be processed
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result of import, SUCCESS if ok, ERROR followed by a message in case of error

6.12.2 AxVFPCompare

STRING AxVFPCompare (RESOURCE InputResource, RESOURCE InputResource2, INT32 CLENGTH, INT32 RLENGTH, FLOAT MINBER, INT32 TIMEPOS)

Version: 1.0
Description: Compares two video fingerprint files and returns the lowest BER and time position.
Parameter List
Name: InputResource
Description: The candidate object to compare
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: fingerprint
Resource Format: video
Ranges:
Name: InputResource2
Description: The reference object to compare
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: fingerprint
Resource Format: video
Ranges:
Name: CLENGTH
Description: Nr. of Subfingerprints in the candidate Object
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: RLENGTH
Description: Nr. of Subfingerprints in the reference Object
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Name: MINBER
Description: Minimal BER found (in percent)
Paramater Type FLOAT
Default Value:
Constraints:
Resource Type:
Ranges:
Name: TIMEPOS
Description: Subfingerprint position for the minimal BER found
Paramater Type INT32
Default Value:
Constraints:
Resource Type:
Ranges:
Result: Result
Result type: STRING
Result Description: The result the comparison, SUCCESS if ok, ERROR followed by a message in case of error

6.13 Plagiarism

Category: ContentProcessing
Identifier: Plagiarism
Library: Fingerprint **Version:** 0.2.0
Vendor: Axmedis
Main Library: plagiarismplugin

Description: Plugin for plagiarism detection

FunctionList:

- Compare

6.13.1 Compare

STRING Compare (RESOURCE Source, RESOURCE Target)

Version: 1.0
Description: Compare two text documents
Parameter List
Name: Source
Description: The source document
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:
Name: Target
Description: The target document
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:
Result: Result
Result type: STRING
Result Description: The result of evaluation, SUCCESS if ok, ERROR followed by a message in case of error

6.14 LanguageGuesser

Category: ContentProcessing
Identifier: LanguageGuesser
Library: Descriptor **Version:** 1.001
Vendor: Axmedis
Main Library: languageguesserplugin.dll

Description: Plugin for extracting the language of a document

FunctionList:

- LanguageGuesser

6.14.1 LanguageGuesser

STRING LanguageGuesser (RESOURCE InputResource, STRING Language)

Version: 1.0
Description: Retrieves the main language of the document.
Parameter List
Name: InputResource
Description: The Resource to be processed
Paramater Type RESOURCE
Default Value:
Constraints:
Resource Type: text
Resource Format: plain
Ranges:
Name: Language
Description: A string representing the language (one of {de, en, es, fr, it} values)
Paramater Type STRING
Default Value:
Constraints:
Result: Result
Result type: STRING
Result Description: The result of detection, SUCCESS if ok, ERROR followed by a message in case of error