Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE3.1.2D
Framework and Tools Specifications (Fingerprint and Descriptors)

Version: 2.1
Date: 15/03/2005
Responsible: FHGIGD (supervision of DSI)

Project Number: IST-2-511299 Project Title: AXMEDIS Deliverable Type: Report Visible to User Groups: No Visible to Affiliated: No Visible to the Public: No.
Deliverable Number: DE3.1.2 Part D Contractual Date of Delivery: January 2005 Actual Date of Delivery: 15 March 2005 Title of Deliverable: Document Work-Package contributing to the Deliverable: WP3.1 Task contributing to the Deliverable: WP3, WP2 Nature of the Deliverable: report Author(s): FHGIGD, DIPITA, EPFL, DSI

Abstract: The integration of content description algorithms used for efficient searching and browsing and verification of AXMEDIS objects are described in this document. It explains the plug-ins mechanisms and the API interface for clients. Also the content processing and description algorithm profiles are defined in this document.
Keyword List: content description algorithms, content identification, fingerprinting, meta data, MPEG-7, plug-in, algorithm profiles

AXMEDIS Copyright Notice

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

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

1. **DEFINITIONS**

   i. "**Acceptance Date**" is the date on which these terms and conditions for entering into possession of the document have been accepted.

   ii. "**Copyright**" stands for any content, document or portion of it that is covered by the copyright disclaimer in a Document.

   iii. "**Licensor**" is AXMEDIS Consortium as a de-facto consortium of the EC project and any of its derivations in terms of companies and/or associations, see www.axmedis.org

   iv. "**Document**" means the information contained in any electronic file, which has been published by the Licensor's as AXMEDIS official document and listed in the web site mentioned above or available by any other means.

   v. "**Works**" means any works created by the licensee, which reproduce a Document or any of its part.

2. **LICENCE**

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

3. **TERM AND TERMINATION**

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

4. **USE**

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

5. **COPYRIGHT NOTICES**

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

6. **WARRANTY**

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

7. **INFRINGEMENT**

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

8. **GOVERNING LAW AND JURISDICTION**

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

Please note that:

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

Table of Content

1 Executive Summary and Report Scope

The full AXMEDIS specification document has been decomposed in the following parts:
A. general aspects up to the description of the content model
B. Viewers and players, including plug ins, etc.
C. Content Production tools and algorithms
D. Fingerprint and descriptors algorithms and tools
E. Database area, query support and Content Crawling from CMS
F. AXEPTool area, for B2B distribution and Programme and Publication for B2C distribution
G. Workflow aspects and tools
H. Protection tools and support, Certification and Supervision and Accounting tools
I. Distribution tools and AXMEDIS Portal
J. Definitions, tables, terminology, acronyms, lists, references, links and Appendixes

This document contains Part D only.

Within AXMEDIS content descriptors are used for the verification of AXMEDIS objects (fingerprints or perceptual hashes) and for improved content search and browsing technology (human understandable content descriptors). The different types of content descriptors considered within AXMEDIS are low-level and high-level descriptors.

Within this deliverable we describe the different interfaces how content description and content adaptation algorithms are integrated into the different components within AXMEDIS: Plug-Ins for AXOM, Plug-Ins for the Collector Engine and API for Clients.

A general profile for content description and algorithm is defined. This profile allows an efficient management of the developed plug-ins. The plug-in technology was chosen to guarantee the requirements of the users: extensibility of the available functionality by functions depending on their needs.

In the MPEG-7 standard, which is formally named “Multimedia Content Description Interface”, a descriptor definition language is defined. This is considered in the profile description for the different content description algorithms. This also document describes this relationship.

The algorithms considered so far in this document comprise the following content description algorithms:

- Low- and high-level descriptors for meta data extraction (partially human readable) for efficient content search and browsing and low-level descriptors for fingerprinting (AXMEDIS objects verification) of
    - text documents,
    - audio files,
    - video files, and
    - general digital resources.

The scheme definition considered different available algorithms and allows the easy integration of different content description algorithms for different media types.

2 Fingerprint Extractors Area (FHGIGD)

The Fingerprint extractor area includes algorithms and tools for the calculation of content descriptors and fingerprints. Content descriptors describe multimedia content. These descriptions can be computer as well as human understandable. Thus content descriptors are very important for searching and browsing information. The relevance of content descriptors resulted in the MPEG-7 standard that is formally named "Multimedia Content Description Interface" [MPEG-7]. The aim of MPEG-7[1] is to provide "a rich set of standardized tools to describe multimedia content". Due to the huge amount of data processed and exchanged automatically within the AXMEDIS framework, the content descriptors have to be calculated automatically. The relevant algorithms for this task are defined in the Fingerprint Extractors Area.

Generally content descriptors summarize characteristics of the content. Within the software applications different types of content descriptors can be distinguished:

- A **hash value** is calculated by a hash function that "converts an input from a (typically) large domain into an output in a (typically) smaller range (the hash value, often a subset of the integers). … A good hash function is one that experiences few hash collisions in the expected domain of strings it will have to deal with; i.e. it would be possible to uniquely identify most of these strings using this hash." [WIFS]

- **Low-level descriptors** describe digital information a very basic level. They can be extracted automatically (at least most of them). However, these descriptors are not always clear to humans. For example an energy distribution within an audio signal is more difficult to understand for humans than a melody.

- **Digital fingerprints** (or **perceptual hash values**) are a specific kind of low-level descriptors. They can be compared with human fingerprints: One the one hand they are (almost) unique identifiers.

- **High-level descriptors** are easy to understand for humans. E.g. the melody of a audio signal is a high level description. In contrast to low-level descriptors they require (much) more human interaction. This is clear as the description of a video sequence or its summary requires an understanding of the input data and also context information.

Within the AXMEDIS framework different applications for content descriptors can be identified:

- **Identification of similar content** is a desired functionality if a huge amount of data is stored in a database, which is the case in AXMEDIS. Thus, descriptors – preferably high-level descriptors – support the exploration of and the browsing in unknown content.

- **Identification of a single piece of content** can be interesting when content is only partially accessible or the identification of the content is not directly possible. E.g. for an audio segment the corresponding song has to be identified. This cannot be done by using hash values as the results of the hash functions are sensitive to bit manipulations. Thus, cropping strongly affects the resulting hash value. Fingerprints (also called perceptual hash values) are developed for this purpose. However, within AXMEDIS this is only a minor application for fingerprints. More important is the following one.

- **Authentication of the content** is necessary to ensure that only valid content is distributed on the AXMEDIS framework. AXMEDIS objects can consist of several different resources. To ensure the validity of an AXMEDIS object technical methods have to be developed and implemented, which

[1] Within MPEG-7 content descriptors are also called „features".

allow validating AXMEDIS objects. Typically hash values are used for this. However, they are sensitive for bit changes. If content adaptation is performed hash values cannot be used for validation of the content. Thus more sophisticated descriptions are needed. Typically specific low-level descriptors are used for this purpose, which are consequently called fingerprints or perceptual hash functions (cf. above).

The following table shows the content descriptors suitable for the individual applications. Obviously high level descriptors are advantages in comparison to low-level descriptors. However, only very few of them can be calculated automatically. Sub-optimal solutions are indicated by '(X)' in the following table.

	Authentication and integrity of the AXMEDIS object	**Identification of a single piece of content**	**Identification of similar content**
Hash functions	X	(X)	
Low level descriptors		(X)	X
Fingerprints/ perceptual hash functions	X	X	(X)
High level descriptors		(X)	X

The different algorithms for calculating the content descriptors are virtually grouped in the **Pool of Fingerprint Algorithms**. This virtual pool contains and manages the content descriptor algorithms available locally. The content descriptor algorithms are stored in libraries. New algorithms can be easily added through plug-in technologies without recompilation of the program. This allows content owners and distributors to integrate their preferred or already used technology within the AXMEDIS framework.

The calculation of the content descriptors can be initialized from different AXMEDIS components. The components, which have access to the Pool of Fingerprinting Algorithms are:

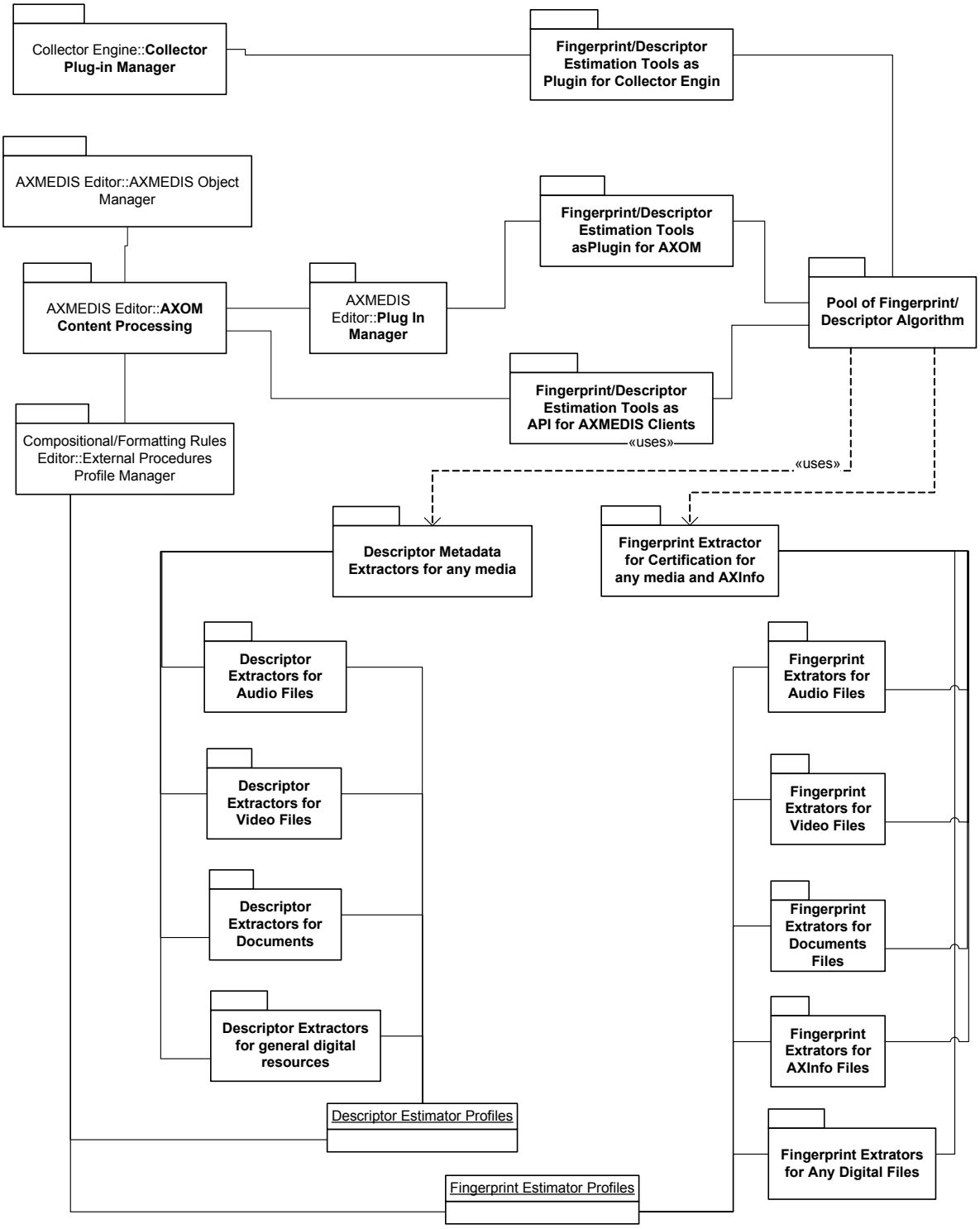
- Fingerprint Estimation Tools as **Plug-ins for Collector Engine**: The collector engine automatically imports content from the local CMS of content owners and distributors into the AXMEDIS database. This requires the automatic calculation of metadata during respectively immediately after the import to ensure content accessibility.

- Fingerprint Estimation Tools as **Plug-ins for AXMEDIS Editors and Engines**: Besides the import of existing content, content can also be produced using AXMEDIS Editors or AXMEDIS engines. Also for new content the calculation of content descriptors has to be possible. The functionality is provided to the components via the AXOM.

- Fingerprint Estimation Tools as **API (LIB) for AXMEDIS Clients** provide the possibility to verify content received by the end-users/customers.

These different components and their relationship are shown in the following diagram. There the content description algorithms are grouped according to their main functionality within AXMEDIS and their (input) content types. The vertical segmentation visualizes the two man functionalities of content description and authentication. The horizontal segmentation indicates the content types relevant within AXMEDIS: audio, video, documents, and general digital resources (and AXInfo files).

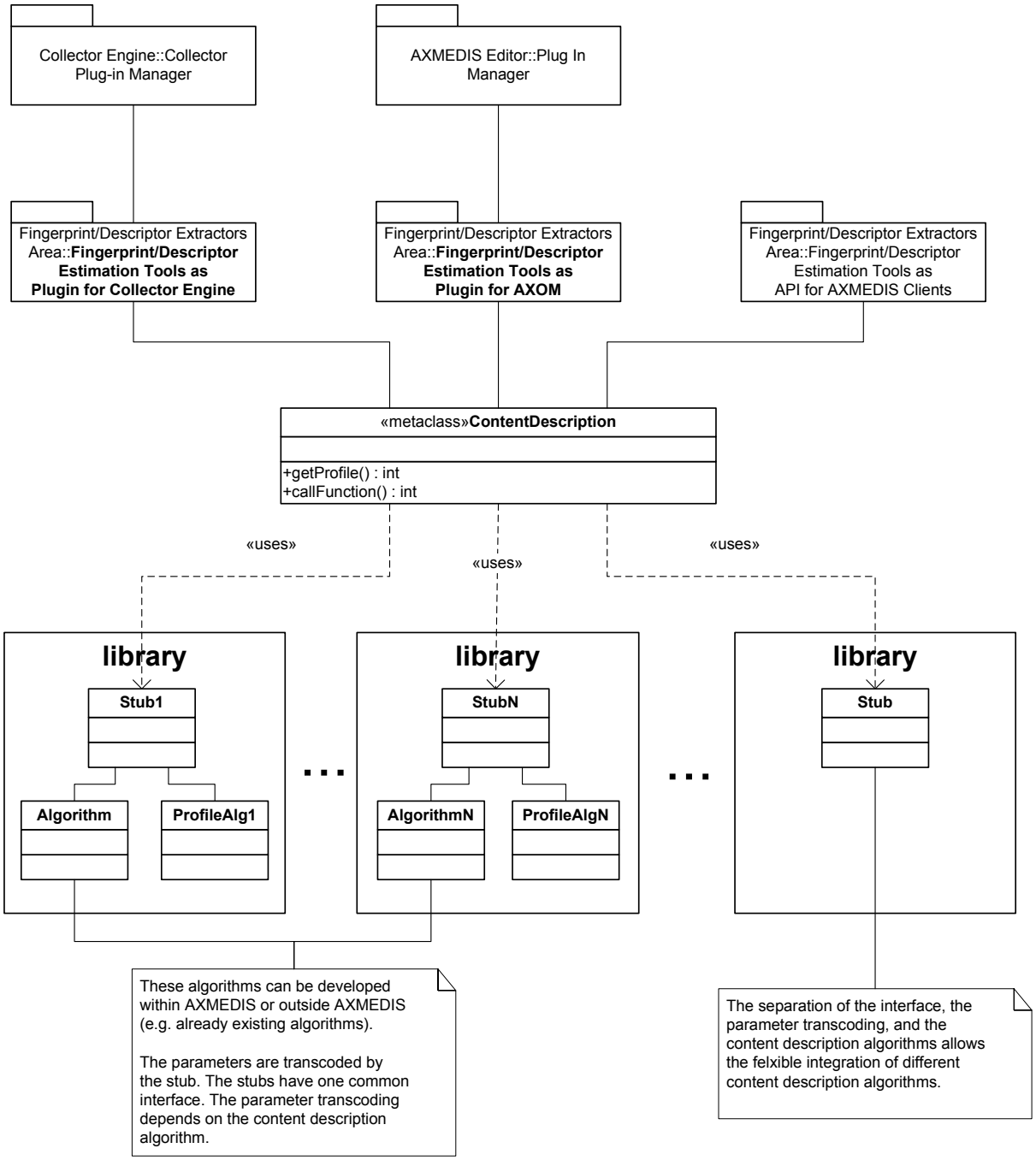
Fingerprint/Descriptor Extractors Area



The above diagram shows the different categories of content description and content fingerprinting algorithm. It is not an implementation view, as e.g. from an implementation point of view there is no distinction between content descriptors and fingerprinting algorithms. Similarly, the pool of fingerprinting/content description is a logical aggregation. (The implementation view is shown in the next

figure.) The different available algorithms are handled by the AXMEDIS editor plug-in manager. The plug-in manager main functionality is based on the content descriptor estimation profiles. These profiles are human and computer readable descriptions of the algorithms and their parameters. They are relevant for users to ensure the functionality of the different algorithms. Also they are used within the AXMEDIS framework as information e.g. for validation of workflow operations (verification of input and output formats). They are described in detail in the next section and in DE3.1.2 Part C (Content Production) in section 2.5.7. Here, we focus on the implementation aspects of the content descriptors.

The previously shown vertical and horizontal separation according to the functional aspects and content types is only an logical view on the Fingerprinting Extractors Area. In the implementation this separation does not exist. In contrast for the different algorithms one single interface will be developed. This interface allows existing and future algorithms to be plugged in easily as shown in the next figure.



The involved components and modules are:

- The plug-in managers (Collector Engine Plug-In Manager and AXMEDIS Editor Plug-In Manager) manage the available content description algorithms. The functionality of new algorithms are queried by the getProfile()-method defined in the interface ContentDescription. New algorithms are registered. The function calls are transferred to the corresponding algorithm.

  Note: During writing this specification it indicated that the AXMEDIS Editor Plug-In Manager and the Collector Engine Plug-In Manager merge into one general Plug-In Manager component (see also below).

- The stub has to be implemented for each algorithm. It handles the registration of new algorithms and the call of the library functions. Furthermore it handles the profile management. For each algorithm a human and a system readable description (profile) of the algorithms interface is required. This description profile is stored as an XML-object as part of each library. This allows an easy exchange of the description profile as an string object with requesting modules.

- The stub receives the general function call (stored in an XML-object) and traduces it to the specific function. Thus, each algorithm requires its individual stub for the parameter conversion. A general stub will be provided to show how it has to be implemented.

- The stub calls the specific content description algorithm and traduces the result back (XML). The conversion of the return parameters is symmetric to the conversion of the input parameters.

Due to limited resources and performance the AXMEDIS clients directly call the functionality they require without involving a plug-in manager.

The different descriptors are metadata within the AXMEDIS objects. They can be used for technical queries. This results in some implications for the database structures especially for the depending search and retrieval algorithms used within the AXMEDIS database. This is caused by the partially complex structures of the calculated fingerprints and metadata descriptors.

2.1 Fingerprint Estimation Tools as Plug-ins for AXOM (FHGIGD)

2.1.1 Overview

The Fingerprint Estimation Tools as **Plug-ins for the AXOM** is accessible in different ways/for different purposes including:

- a user of the editor after (s)he created content manually,
- the content production tool when new content is created automatically
- the content formatting tool when content is formatted automatically
- the AXEPTool to verify the content entering or leaving the P2P exchange network

The AXOM calls the content description functionality via the AXOM Content Processing. The AXOM Content Processing uses the content description functionality through the Plug-In Manager. This is shown in the following figure.

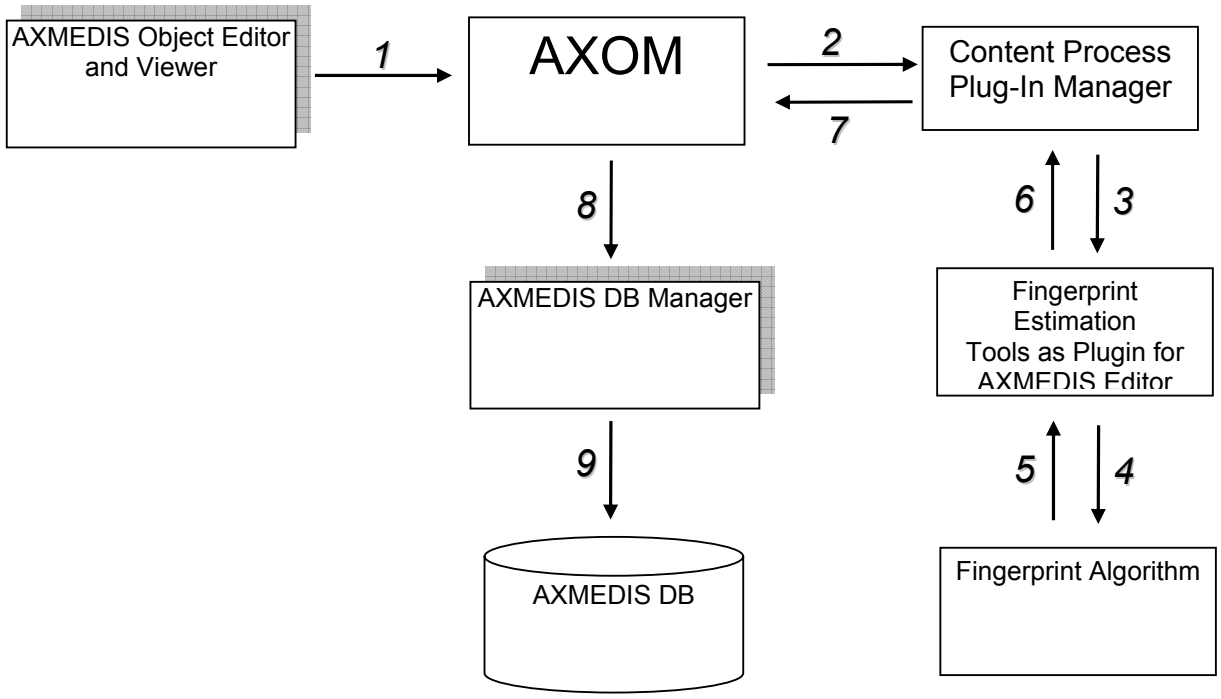


Figure: Estimating Fingerprint: AXMEDIS Editor

An example for using the content description/fingerprinting via the AXOM for monitoring the the uploads to and the downloads from the AXPETool is shown in the next figure.

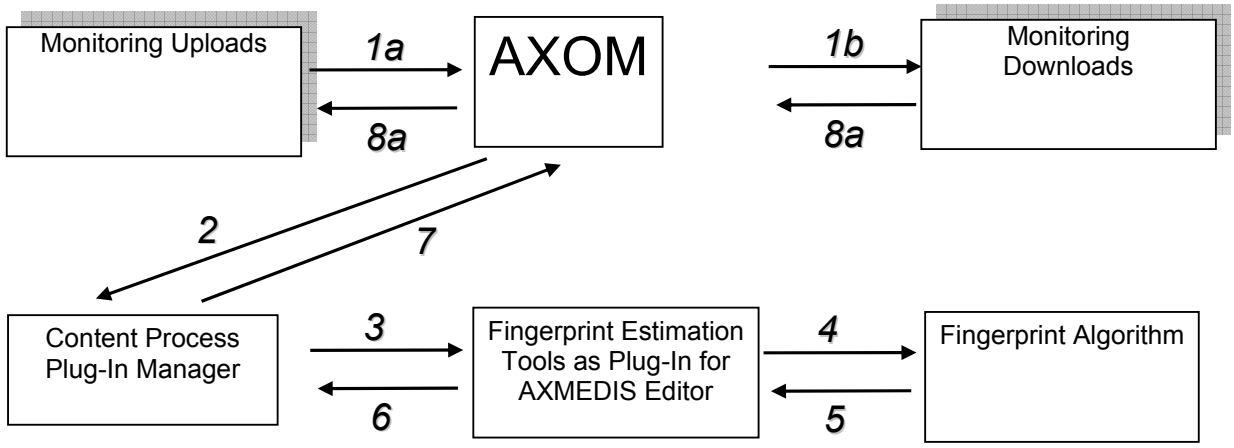


Figure: Using fingerprint for verification of object consistency

Content adaptation algorithms as well as content description estimation and calculation algorithms can be dynamically plugged in into the AXMEDIS framework. To allow this the AXMEDIS content processing plug-ins are stored in a specific directory. Upon their initialisation the AXMEDIS plug-in managers (AXMEDIS Editor::Plug-In Manager and the Collector Engine::Collector Plug-In Manager) scan this specific directory. The AXMEDIS plug-in managers load the available libraries and check their functionality by loading the libraries description.

This description contains all relevant information about the algorithm available in the library. As content adaptation and content description algorithm are strongly related a common XML Schema was developed for for the different plug-ins within AXMEDIS. It allows the description of all kinds of algorithms, which can be

plugged in the AXMEDIS framework. The general aspects of the plug-in architecture and description is described in DE3-1-2C (Framework and Tools Specifications – Viewer and Players).

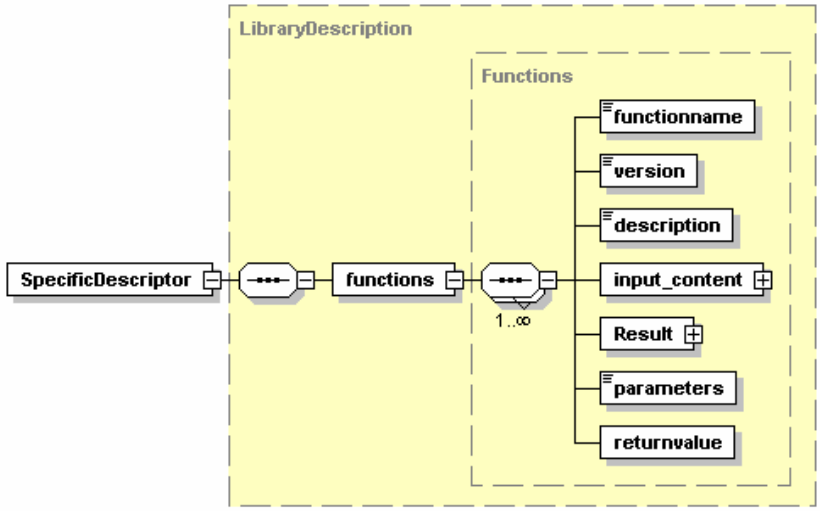
The plug-in description consists of

- a general description of the library, containing general information about the plug-in like vendor, version, or a human readable description;
- components signatures (the signatures of the entire plug-in),
- a specific descriptor, which describes the specific features of the plug-in, and
- a signature of the whole profile without the signature elements itself.

The specific descriptor of the plug-in features depends on the individual function provided by the plug-in.

As content adaptation and content description algorithms can be distinguished but are closely related for content description and content adaptation a unique XML schema was defined. Further information about content adaptation algorithms is available at DE3-1-2C (Framework and Tools Specification – Content Production).

A profile of a content adaptation/description plug-in must contain at least one function. Thus, the description contains information about all the available functions. Each function description comprises seven parts, as shown in the figure below:



1. Function name: the name of the function.
2. Version: the version of the function
3. Description: a human readable description of the functionality implemented. This is also intended a short help for the user.
4. Input content: information about the content that is processed by the content adaptation algorithm.
5. Result: the results produce/calculated by this function
6. Parameters: information about the parameters needed
7. Return value: information about the status of the function after processing information.

Content adaptation and content description algorithms process different kinds of content. Therefore, the description includes the type and the format. The type corresponds to the MIME (Multipart Internet Mail Extensions) type while format corresponds to the MIME subtype. Information about the MIME content-types can be found in the RFC 2045 [RCF2045], 2046 [RCF2046] and 2077 [RCF2077].

Content adaptation and content description differ in the result returned:

- Content adaptation algorithms process content and return (typically the same) content again. This is considered in the result output content, which again comprises type and format (as above corresponding to MIME type and subtype).

- Content description algorithms return descriptors. However, the description of parameters itself must be a description of parameters. Therefore not a fixed structure is proposed. Within AXMEDIS the format of these descriptors is MPEG-7 compliant. In MPEG-7 a Description Definition Language (DDL) is standardised (ISO/IEC JTC1/SC29/WG11 – Part 2, see Appendix Content Descriptors Description).

The resulting XML schema for content adaptations and content description algorithm is in Part B in the section "Plug-in function description".

The plug-ins are accessed via the plug-in manager. For the AXMEDIS Editor Plug-In Manager the corresponding part to the stub has to be implemented. This module is accessed via the AXMEDIS Editor Plug-In Manager and calls the corresponding plug-in functionality.

Module Profile		
Fingerprint/Descriptor Estimation Tools as Plugin for AXOM		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multi	
Language of Development	C++	
Responsible Name	Martin Schmucker	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDIS Editor::Plug-In Manager		
File Formats Used	Shared with	File format name or reference to a section
None	None	
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
None		

As described before, for each plug-in a stub has to be implemented. The following module describe this general stub.

Module Profile		
Fingerprint/Descriptor Estimation Tools (Stub)		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multi	
Language of Development	C++	
Responsible Name	Martin Schmucker	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDIS Editor::Plug-In Manager (via the corresponding Plug-In)		
File Formats Used	Shared with	File format name or reference to a section
None	None	
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	Native Code: C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
None		

2.2 Fingerprint Estimation Tools as Plug-ins for the Collector Engine (WP4.2.2: FHGIGD)

The Fingerprint Estimation Tools as **Plug-ins for the Collector Engine** are called during the transfer of content from the CMS to the AXDB. The collector engine initiates the calculation of the content descriptors. The content descriptors are calculated by the available content description algorithms. This is shown in the following figure. Here, the plug-in mechanism in the collector engine allows an easy integration of existing content description algorithms.

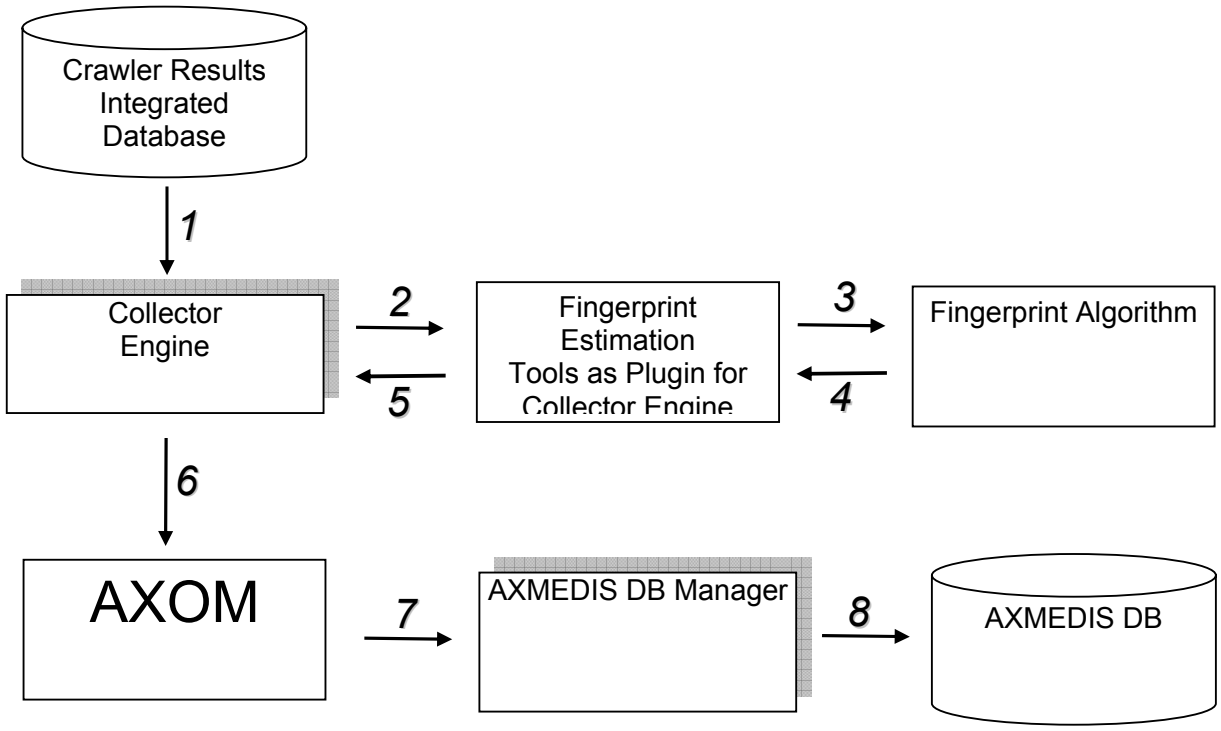


Figure: Estimating Fingerprint: Collector Engine

The functionality as well as the logic are the same as for the AXMEDIS Editor Plug-In Manager. However, due to the different implementations (FOCUSEEK is already implemented with its own plug-in interface) different interfaces are needed.

Note: During writing this specification it indicated that the AXMEDIS Editor Plug-In Manager and the Collector Engine Plug-In Manager merge into one general Plug-In Manager component. If this is the case the following subsections in these paragraphs are obsolete and only the previous description of the AXMEDIS Editor Plug-In Manager is valid.

2.2.1 Collector Engine Plug-In Profile/Interface

The Collector Engine Plug-In interface is described in Part C. Here, only a short a summary is given.

Three different types of plug-ins are supported by the Crawler system:

- **Protocol plug-ins** receive an URL specifying a document and retrieve it.
- **Parser plug-ins** receive the original document contents and extract information from it.
- **Mission plug-ins** coordinate the retrieval process.

The content description and content adaptation functionality parses content for processing it. Thus they have to be implemented as parser plug-ins.

Similar to the previous described plug-ins for the AXOM the plug-ins are separated in two parts: The functionality is called from the crawler-plug in. And the implementation of the functionality is separated.

The same unique interface is defined to reduce the programming efforts and to increase the compatibility between the crawler and the AXOM plug-ins.
Here, only the functionality required in the Collector Engine-Plug-In is specified. As described before, for each plug-in a stub has to be implemented. The specification of the stub is available in the previous section.

Module Profile		
Fingerprint/Descriptor Estimation Tools as Plugin for Collector Engine		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
Collector Engine:: Collector Plug-in Manager		
File Formats Used	Shared with	File format name or reference to a section
None	None	
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

2.3 Fingerprint Estimation Tools as API (LIB) for AXMEDIS Clients (FHGIGD)

Due to limited resources and performance the AXMEDIS clients directly call the functionality they require without involving a plug-in manager.

Not all fingerprint estimation tools will be available for all AXMEDIS clients. The different AXMEDIS clients have limitations in performance or available resources. Thus, calculation of computational complex fingerprints is not feasible on AXMEDIS clients with lower-performance processing units (like current mobile phones).

Nevertheless, the calculation of check-sum should be available as minimal functionality on each client.

Instead of implementing a plug-in manager the functionality provided by the stub is directly accessed. This stub is described in the section General Content Description Profile and Algorithm below.

Module Profile		
Fingerprint/Descriptor Estimation Tools as API for AXMEDIS Clients		
Executable or Library(Support)	Library	
Single Thread or Multithread	Singlethreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDIS Editor:: AXOM Content Processing		
File Formats Used	Shared with	File format name or reference to a section
None	None	
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	None
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
None	None	None

2.4 General Content Description Profile and Algorithm (FHGIGD, DIPITA, DSI)

All functionality provided by content description and adaptation algorithms are accessed via a unique interface. This is the super-class for the stubs to be integrated. The corresponding plug-ins have to implement the functionality described below:

- **`int GetProfile(string profile)`**: This method returns the plug-in specific profile of the functions available in the plug-in.
    - "**profile**" is an XML-description according to the previously described schema.

- **`int GetProfile(string functionName, tData data, tParameterList parameters, tResult result)`**: This is the general function call for the functions available in the plug-in:
    - "**functionName**" determines the function called in the plug-in
    - "**data**" is the input content that will be processed. The data is given as a binary object. Specific conversion or access functionality (e.g. streaming for audio or video files) has to be provided within the stub if required.
    - "**parameters**" is a list of tuples. The tuples consists of parameter name (type string) and the parameter value (type binary). Parameter conversion has to be done by the stub.
    - "**result**" is the result of the function call. For content adaptation algorithm this is the adapted (multi-media) object. In the case of content description algorithm the result is a list of tuples. Each of the tuples consists of a descriptor name and the calculated value. The stub has to process and convert the results from the called library.

The following interface is derived:

```
module SpecificDescriptor  {

   interface ContentDescriptionAdaptation {

      /**
       * tData: (Multi-media) content is given as a byte object.
       */
      typedef sequence<octet> tData;


      /**
       * tParameterList: The input parameters are given as a parameter list.
       *                 Each entry in this list is a tuple(name, value).
       */
      struct tParameter {
         string          parameterName;
         sequence<octet> parameterValue;
      };

      typedef sequence <tParameter> tParameterList;



      /**
       * tDescriptorList: The input parameters are given as a parameter list.
       *                  Each entry in this list is a tuple(name, value).
       */
      struct tDescriptor {
         string          descriptorName;
         sequence<octet> descriptorValue;
      };

      typedef sequence <tDescriptor> tDescriptorList;



      /**
       * tResult: The result is either a list of descriptors or a adaptet content.

       */
      union tResult switch (boolean)
      {
        case TRUE:  tDescriptorList resultDescriptor;
        case FALSE: tData resultData;
      };
```

```
        /**
         * getProfile returns the xml description of the profile as a string.
         */
        short getProfile(out string profile);



        /**
         * callFunction is the unique interface to the stub.
         */
        short callFunction(  in string functionName,
                             in tData data,
                             in tParameterList parameters,
                             out tResult tResult);
    };

};
```

This functionality has to be provided to the AXMEDIS Editor Plug-In Manager and the AXMEDIS client by the following module:

Module Profile		
Content Description		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
Collector Engine::Collector Plug-in Manager AXMEDIS Editor::Plug-In Manager (via the corresponding Pllug-In) AXMEDIS Editor::AXOM Content Processing		
File Formats Used	Shared with	File format name or reference to a section
None	None	None
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	None
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK,

2.5 Descriptor/M

2.5.1 Descriptor/M

Given a text in any su
format. Texts have to b
the tokenization, the do

distribution and reference statistics. The range of identifiable "language" metadata is English (en), French (fr), German (de), Italien (it), and Spanish (es).

3. Pre-processing

- *Disambiguation of the periods:* The identification of periods that have a full stop value is needed to tokenize the document in relevant linguistic units.
- *Text tokenization*: Different units of tokenization: words, chunks, periods.
- *Lemmatizatio:*. Each word receives a tag regarding the part of speech (PoS) and the lemma. To be accomplished by an external multilingual tagging tool (e.g. TreeTagger, Xerox Tagger, Connexor Machinese) for de, en, fr, it.
- *Selection of nouns*: From the results of lemmatization, only nouns are assumed to be potential keyword/descriptors of the document. Output of the document-specific nouns frequency list.

Algorithm A

4A. Statistic Processing

- *Selection of the n% most frequent nouns*
- *Statistic clustering.* The clustering process is based on two co-occurrence measures of the output items of the previous step (co-occurrence is estimated within period contexts): similarity-based clustering (terms with similar distribution of co-occurrence with other terms even if they are not frequent, measured by the Jensen-Shannon divergence); pairwise clustering (terms that co-occur frequently in the same period contexts, measured by mutual information score)
- *Keyness value estimation.* Estimation of the importance of the pre-selected words of a document. Various algorithms can be used: χ^2, log-likelihood, z-score. The estimation generates a score for the keyword candidates.

5A. Fine grained collocation extraction.

Analysis of the strict co-occurrence (in contiguous context of fixed length) of each selected keyword with other related words. If the co-occurrence overruns a fixed threshold (to be experimentally detected) the keyword become a complex keyword.

6A. Output of the document keyword

Selection of the words (or multi-words) with the highest score.

Algorithm B

4B. Statistic comparison with general corpus frequency lists.

The frequency list extracted from a specific document is compared with a general one, extracted form a big corpus. The more prominent differences (mostly within the nouns) are taken into account as a fist step to recognize the document specific lexicon, that is assumed to contain the relevant keyword candidates. In principle, the result of this process can be also compared (and refined) with the internal statistic scoring discussed at point 4A.

5B. Fine grained collocation extraction.

Analysis of the strict co-occurrence (in contiguous context of fixed length) of each selected keyword with other related words. If the co-occurrence overruns a fixed threshold (to be experimentally detected) the keyword become a complex keyword.

6B. Output of the document keyword

Selection of the words (or multi-words) with the highest score(s).

Algorithm C

4C. Statistic comparison with general corpus frequency lists.

The frequency list extracted from a specific document is compared with a general one, extracted form a reference corpus. The more prominent differences (mostly within the nouns) are taken into account as a fist step to recognize the document specific lexicon, that is assumed to contain the relevant keyword candidates. The result of this process can be compared (and refined) with the internal statistic scoring discussed at point 4A.

5C. Output of the keyword candidates.

Wide selection of the words (or multi-words) with the highest score(s).

6C. Semantic processing

- *Word Sense Disambiguation (WSD):* Words can be ambiguous about their meaning. With respect to this point, in WordNet a word can be associated with one or more concepts, that describe the possible meanings of a single lemma. WSD strategies take into account the semantic context in which the word occurs. The output is the association of each word with one concept, that makes possible the translation of the selected (key)words into the different languages (associated in the multilingual WordNet).
- *Semantic Similarity estimation (SS):* The Leacock-Chorodow concept similarity measure can be used to measure the semantic similarity between two concepts. This measure is based on the length of the paths between pairs of concepts in the WordNet IS-A hierarchy.
- *Semantic clustering:* Clusters of concepts related to each other with decreasing values of *SS* (up to a predefined lower threshold).
- *Scoring:*
  1. first level: *cluster level score*: This score is defined as the sum of two components:
     - *connectivity*, corresponding to a global measure of the semantic relatedness between all concepts in the cluster;
     - *reiteration*, corresponding to a global measure of the importance of the concepts in the cluster.
  2. second level: *concept level score:* This score is defined in a similar way as the sum of two components:
     - *concept level connectivity*;
     - *concept level reiteration*.

These two measures are combined by a function that produces a unique score for each concept.
Words that correspond to concepts with the highest global score are the best candidates to be document keywords.
The cluster with the higher score can be associated with a descriptor (as the ones in the MultiWordNet Domains). The result is the output of the domain of the document.

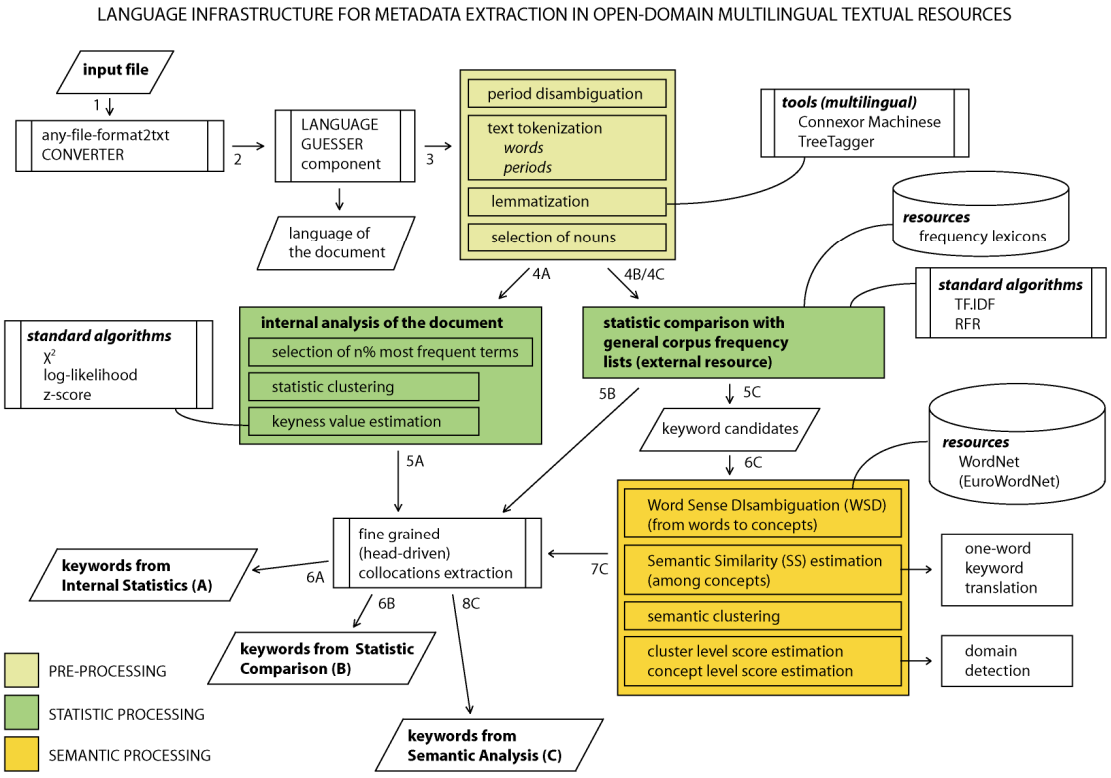
7C. Fine grained collocation extraction.

Analysis of the strict co-occurrence (in contiguous context of fixed length) of each selected keyword with other related words. If the co-occurrence overruns a fixed threshold (to be experimentally detected) the keyword become a complex keyword.

8C. Output of the document keyword

Words (or multi-words) with the highest score(s) are selected and returned.

The components needed for the extraction of metadata from text documents and the information flow in between them is shown in the next figure.



The IDL of the module that contains classes that deal with descriptors extraction is given in the Appendix Interface of the Metadata Extractors for Text.

Within MPEG-7 the following tools are relevant for the meta data extracted from text and documents:

Tool	*Functionality*
Language Identification	Tools for identifying the language of a textual description or of the AV content itself. MPEG-7 uses the XML defined xml:lang attribute to identify the language used to write a textual description.
Text Annotation	Tools for representing unstructured and structured textual annotations. Unstructured annotations (i.e. with free text) are represented using the FreeTextAnnotation datatype. Annotations that are structured in terms of answering the questions "Who? What? Where? How? Why?" are represented using the StructuredAnnotation datatype. Annotations structured as a set of keywords are representation using the KeywordAnnotation datatype. Finally, annotations structured by syntactic dependency relations-for example, the relation between a verb phrase and the subject-are represented using the DependencyStructure datatype.

Language Descriptors defined in MPEG-7:

MPEG-7 distinguishes between the language of the metadata and the language of the content. (However, MPEG-7 has a strong focus on audio-visual content as described below). The xml:lang attribute must be used in the first case-for example, to specify the language in which a textual annotation is written-and the built-in XML Schema language datatype in the second-for example, e.g. to specify the language of an audio track.

```
<!-- ################################################ -->
<!-- Definition of Language Datatype                  -->
<!-- ################################################ -->

<!--- %%% The datatype is already defined in XML      -->
<!--- <simpleType name="language" base="string"/>  -->


<!-- ################################################ -->
<!--   Definition of Classification DS               -->
<!-- ################################################ -->
<complexType name="ClassificationType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="Form" type="mpeg7:ControlledTermUseType" minOccurs="0"/>
            <element       name="Genre"       type="mpeg7:GenreType"       minOccurs="0"
maxOccurs="unbounded"/>
            <element name="Subject" type="mpeg7:TextAnnotationType" minOccurs="0"/>
            <element   name="Purpose"   type="mpeg7:ControlledTermUseType"   minOccurs="0"
maxOccurs="unbounded"/>
            <element   name="Language"   type="mpeg7:ExtendedLanguageType"   minOccurs="0"
maxOccurs="unbounded"/>
            <element      name="SubtitleLanguage"      type="mpeg7:ExtendedLanguageType"
minOccurs="0" maxOccurs="unbounded"/>
            <element    name="ClosedCaptionLanguage"    type="mpeg7:ExtendedLanguageType"
minOccurs="0" maxOccurs="unbounded"/>
            <element name="SignLanguage" minOccurs="0">
               <complexType>
                  <attribute name="primary" type="boolean" use="optional"/>
                  <attribute name="translation" type="boolean" use="optional"/>
               </complexType>
            </element>
            <element name="Release">
               <complexType>
                  <sequence>
                     <element   name="Country"   type="mpeg7:countryCode"   minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
                  <attribute name="date" type="mpeg7:timePointType" use="optional"/>
               </complexType>
            </element>
            <element name="Target">
               <complexType>
                  <sequence>
                     <element      name="Market"      type="mpeg7:ControlledTermUseType"
minOccurs="0" maxOccurs="unbounded"/>
                     <element name="Age" minOccurs="0">
                        <complexType>
                           <attribute         name="min"         type="nonNegativeInteger"
use="optional"/>
                           <attribute         name="max"         type="nonNegativeInteger"
use="optional"/>
                        </complexType>
                     </element>
                     <element   name="Country"   type="mpeg7:countryCode"   minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
               </complexType>
            </element>
            <element      name="ParentalGuidance"      type="mpeg7:ParentalGuidanceType"
minOccurs="0" maxOccurs="unbounded"/>
            <element   name="MediaReview"   type="mpeg7:MediaReviewType"   minOccurs="0"
maxOccurs="unbounded"/>
```

```
        </sequence>
      </extension>
   </complexContent>
</complexType>


<!-- ################################################    -->
<!--   Definition of ExtendedLanguage Datatype           -->
<!-- ################################################    -->
<complexType name="ExtendedLanguageType">
   <simpleContent>
      <extension base="language">
         <attribute name="type" use="optional" default="main">
            <simpleType>
               <restriction base="string">
                  <enumeration value="main"/>
                  <enumeration value="original"/>
                  <enumeration value="other"/>
                  <enumeration value="backgroundOriginal"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </simpleContent>
</complexType>
```

Keywords Descriptors defined in MPEG-7:

```
<!-- ########################################################  -->
   <!-- TextAnnotation Datatype                                    -->
   <!-- ########################################################  -->
   <complexType name="TextAnnotationType">
      <choice maxOccurs="unbounded">
         <element name="FreeTextAnnotation" type="mpeg7:TextualType"/>
         <element name="StructuredAnnotation" type="mpeg7:StructuredAnnotationType"/>
         <element name="DependencyStructure" type="mpeg7:DependencyStructureType"/>
         <element name="KeywordAnnotation" type="mpeg7:KeywordAnnotationType"/>
      </choice>
   </complexType>
   <!-- ######################################################## -->
   <!-- Definition of StructuredAnnotation Datatype           -->
   <!-- ######################################################## -->
   <complexType name="StructuredAnnotationType">
      <sequence>
         <element   name="Who"           type="mpeg7:TermUseType"           minOccurs="0"
maxOccurs="unbounded"/>
         <element   name="WhatObject"       type="mpeg7:TermUseType"       minOccurs="0"
maxOccurs="unbounded"/>
         <element   name="WhatAction"       type="mpeg7:TermUseType"       minOccurs="0"
maxOccurs="unbounded"/>
         <element   name="Where"          type="mpeg7:TermUseType"          minOccurs="0"
maxOccurs="unbounded"/>
         <element        name="When"        type="mpeg7:TermUseType"        minOccurs="0"
maxOccurs="unbounded"/>
         <element        name="Why"        type="mpeg7:TermUseType"        minOccurs="0"
maxOccurs="unbounded"/>
         <element        name="How"        type="mpeg7:TermUseType"        minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute ref="xml:lang" use="optional"/>
   </complexType>
   <!-- ######################################################## -->
   <!-- Definition of KeywordAnnotation Datatype              -->
   <!-- ######################################################## -->
   <complexType name="KeywordAnnotationType">
      <sequence maxOccurs="unbounded">
```

```
         <element name="Keyword">
            <complexType>
               <simpleContent>
                  <extension base="mpeg7:TextualType">
                     <attribute name="type" use="optional" default="main">
                        <simpleType>
                           <restriction base="string">
                              <enumeration value="main"/>
                              <enumeration value="secondary"/>
                              <enumeration value="other"/>
                           </restriction>
                        </simpleType>
                     </attribute>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
      </sequence>
      <attribute ref="xml:lang" use="optional"/>
   </complexType>
   <!-- ######################################################## -->
   <!-- Definition of Dependency Structure Datatype              -->
   <!-- ######################################################## -->
   <complexType name="DependencyStructureType">
      <sequence>
         <element       name="Sentence"       type="mpeg7:DependencyStructurePhraseType"
maxOccurs="unbounded"/>
      </sequence>
      <attribute ref="xml:lang" use="optional"/>
      <attribute name="phonogrammicAlphabet" use="optional">
         <simpleType>
            <union>
               <simpleType>
                  <restriction base="NMTOKEN">
                     <enumeration value="Roman"/>
                     <enumeration value="Kana"/>
                     <enumeration value="Hangul"/>
                     <enumeration value="Pinyin"/>
                  </restriction>
               </simpleType>
               <simpleType>
                  <restriction base="mpeg7:termReferenceType"/>
               </simpleType>
            </union>
         </simpleType>
      </attribute>
   </complexType>
   <!-- ########################################################  -->
   <!-- Definition of DependencyStructurePhraseType datatype      -->
   <!-- ########################################################  -->
   <complexType name="DependencyStructurePhraseType">
      <choice maxOccurs="unbounded">
         <element name="Quotation" type="mpeg7:DependencyStructurePhraseType"/>
         <element name="Phrase" type="mpeg7:DependencyStructurePhraseType"/>
         <element name="Head">
            <complexType>
               <simpleContent>
                  <extension base="string">
                     <attribute name="terms" use="optional">
                        <simpleType>
                           <list itemType="mpeg7:termReferenceType"/>
                        </simpleType>
                     </attribute>
                     <attribute name="identifier" type="ID" use="optional"/>
                     <attribute name="equal" type="IDREF" use="optional"/>
                     <attribute name="type" use="optional">
                        <simpleType>
```

```
                        <union>
                           <simpleType>
                              <list>
                                 <simpleType>
                                    <restriction base="NMTOKEN">
                                       <enumeration value="noun"/>
                                       <enumeration value="pronoun"/>
                                       <enumeration value="adjective"/>
                                       <enumeration value="verb"/>
                                       <enumeration value="adverb"/>
                                       <enumeration value="conjunction"/>
                                       <enumeration value="preposition"/>
                                       <enumeration value="postposition"/>
                                       <enumeration value="article"/>
                                       <enumeration value="interjection"/>
                                    </restriction>
                                 </simpleType>
                              </list>
                           </simpleType>
                           <simpleType>
                              <list itemType="mpeg7:termReferenceType"/>
                           </simpleType>
                        </union>
                     </simpleType>
                  </attribute>
                  <attribute name="baseForm" type="string" use="optional"/>
                  <attribute      name="phonogrammicRepresentation"      type="string"
use="optional"/>
               </extension>
            </simpleContent>
         </complexType>
      </element>
   </choice>
   <attribute name="identifier" type="ID" use="optional"/>
   <attribute name="equal" type="IDREF" use="optional"/>
   <attribute name="operator" use="optional">
      <simpleType>
         <union memberTypes="mpeg7:dependencyOperatorType mpeg7:termReferenceType"/>
      </simpleType>
   </attribute>
   <attribute name="particle" type="string" use="optional"/>
   <attribute name="synthesis" use="optional" default="dependency">
      <simpleType>
         <restriction base="NMTOKEN">
            <enumeration value="dependency"/>
            <enumeration value="coordination"/>
         </restriction>
      </simpleType>
   </attribute>
</complexType>
```

Within the AXMEDIS project a MPEG-7 descriptor especially for text will be developed. This descriptor reflects specific information that are relevant for text resource and that can be extracted automatically. Relevant features, which have to be considered, are:

- size
- character encoding
- number of characters
- number of words
- number of lines
- number of periods
- number of paragraphs
- words

- lines
- periods

Module Profile		
Descriptor Extractors for Documents		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	Native language necessary: C++	
Responsible Name	Marco Fabbri	
Responsible Partner	DIPITA	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, GNU/Linux	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
CDLStub		
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	Native language with a well-defined language independent interface, e.g. CORBA	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
DOCFRAC		LGPL
GNU Ghostscript		GPL
XPDF		GPL
HTMLDOC		GPL
WordNet (Emglish, Italian, Spanish, French, German)		Free for English, proprietary for other languages
TreeTagger		Free for research. Proprietary for commercial use.

The following conversion libraries will be used for text document conversion:
DOCFRAC (http://docfrac.sourceforge.net/)
Conversion Formats

- RTF to HTML
- RTF to TEXT
- HTML to RTF
- HTML to TEXT
- TEXT to RTF

- TEXT to HTML

Uses

- converting many documents at a time;
- active web pages; and
- converting output from Microsoft's Internet Explorer RTF control to HTML.

Platforms

- Windows;
- Linux command line; and
- programming kit (ActiveX and DLL).

DocFrac is free. It is released under the LGPL.

GNU Ghostscript (http://www.cs.wisc.edu/~ghost/)
Ghostscript is the name of a set of software that provides:

- An interpreter for the PostScript (TM) language and the Adobe Portable Document Format, and
- A set of C procedures (the Ghostscript library) that implement the graphics and filtering (data compression / decompression / conversion) capabilities that appear as primitive operations in the PostScript language and in PDF.

Versions entitled "GNU Ghostscript" are distributed with the GNU General Public License.

XPDF (http://www.foolabs.com/xpdf/)
Xpdf is an open source viewer for Portable Document Format (PDF) files. The Xpdf project also includes a PDF text extractor, PDF-to-PostScript converter, and various other utilities.
Xpdf runs under the X Window System on UNIX, VMS, and OS/2. The non-X components (pdftops, pdftotext, etc.) also run on Win32 systems and should run on pretty much any system with a decent C++ compiler.
Xpdf is designed to be small and efficient. It can use Type 1, TrueType, or standard X fonts.
Xpdf is licensed under the GNU General Public License (GPL), version 2.

HTMLDOC (http://www.easysw.com/htmldoc/)
HTMLDOC converts HTML source files into indexed HTML, PostScript, or Portable Document Format (PDF) files that can be viewed online or printed.
The program is free software and is distributed under GPL.

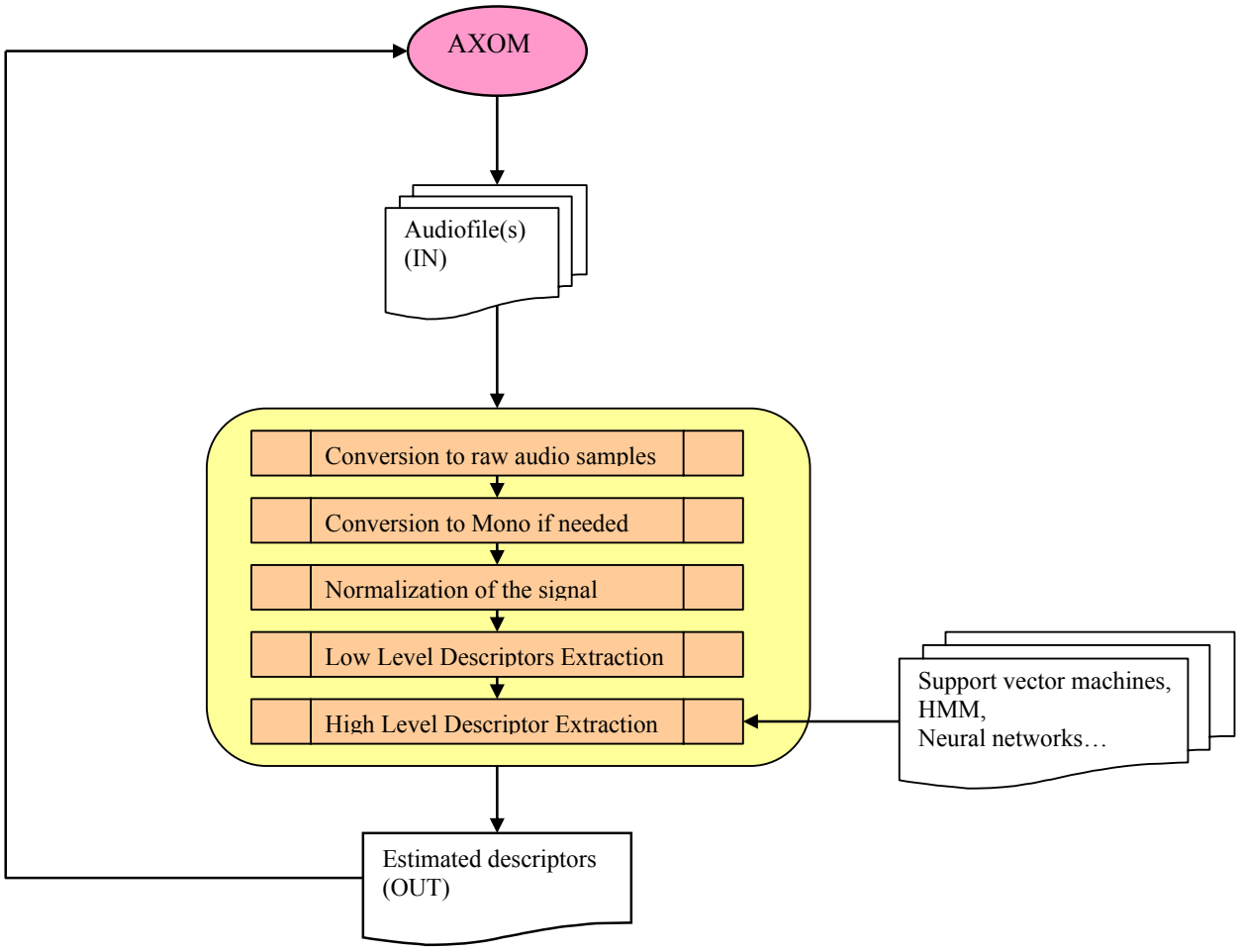
Because some of the used libraries are licensed under GPL, the conversion tool (pool of document converters) will be developed under GPL too. So the conversion tool will be accessed by the plug-in as a separated executable program, not as a library. The communication mechanism between the plug-in and the converter has to be defined.

Language resources needed:
Wordnet (http://wordnet.princeton.edu/)
WordNet® is an electronic lexical database in which nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept.
WordNet is distributed under an “AS-IS” license and can be used in commercial applications without restrictions.
For non-English languages, ELDA (http://www.elda.org/) distributes comparable databases under research and commercial licenses. A license for research will be provided by DSI.

TreeTagger (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html)
The TreeTagger is a tool for annotating text with part-of-speech and lemma information which has been developed within the TC project at the Institute for Computational Linguistics of the University of Stuttgart. The TreeTagger has been successfully used to tag German, English, French, Italian, Greek and old French texts and is easily adaptable to other languages if a lexicon and a manually tagged training corpus are available.
TreeTagger is freely available for research, education and evaluation. Commercial licenses have to be defined contacting the author Helmut Schmid (schmid@@ims.uni-stuttgart.de).

Descriptors extractor plug-in algorithms need comparison data (frequency lists, n-grams statistics, Word-Net synsets) to produce output. It could be reasonable to store such data in an external repository as a DB. In this case the plug-in should have access (read-only) to this DB via a standard communication protocol (ODBC).

2.5.2 Descriptor/Metadata Extractors from Audio Files (EPFL, FHGIGD …..



Descriptor/Metadata Extractor Prototype for Audio Files		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithread	
Language of Development	C++	
Responsible Name		
Responsible Partner	EPFL	
Status (proposed/approved)		
Platforms supported	Microsoft Windows / Linux	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
		Library direct call
File Formats Used	Shared with	File format name or reference to a section
Audio formats (wav, aiff, mp3…)		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
CLAM (?)	0.7	GPL
Torch3 (?)		BSD
LibSVM	2.71	LGPL
Libsndfile	1.0.11	LGPL

The Descriptor/Metadata extraction from audio files area aims at automatically describing music/audio content. This description can be roughly divided into two levels:

- Low Level Description: fundamental frequency, amplitude, spectral deviation, spectral centroid, harmonicity…
- High Level Description: instrument type, singer type, musical form, musical genre, musical style, subjective mood, sound mix characteristics, expressivity features, key, tempo…

The lower level refers to some physical characteristics of the music/audio content that may not be directly interpreted by humans while the higher level tends to describe music/audio content in terms of perception, formal structure and expressiveness.

Some low level features of music/audio content need to be extracted in the context of AXMEDIS as they will be used to evaluate some higher level descriptors. Furthermore, fingerprints of audio files may be computed based on a preliminary low level analysis of the audio signal. Actually, as low level analysis of the signal is the first step in evaluating most descriptors/fingerprints, care will have to be taken to avoid duplicating computations of low level descriptors.

Concerning higher level descriptors, a musical genre extractor should be a priority implementation as musical genre are commonly used to structure large collections of music. The genre taxonomy to be used could be derived for example from the taxonomy used by the All Music Guide (http://www.allmusic.com).

As a side product of musical genre extraction will come some structural descriptors including rhythmic descriptors (tempo, meter, beats), melodic descriptors (tessitura, melody contour, key) and harmonic descriptors (chords changes positions, chord patterns). These descriptors can be expressed in terms of MPEG-7 descriptors and description scheme (AudioTempoDS, AudioRhythmicPatternDS, AudioChordPatternDS, MelodyContourDS …).

These structural descriptors could be used to find some intelligent segmentation points of the musical excerpt. Next step is to produce automatically some summary of the excerpt or at least to point out the most representative parts of the excerpt.

MPEG-7 also proposes some description scheme for instrument timbre (where the timbre is defined as the perceptual features that make two sounds having the same pitch and the same loudness sound different). These descriptors aim at describing isolated sounds and are targeted towards authoring tools for sound designers or musicians or retrieval tools for producers.

One may also be interested in having perceptual descriptors of the global timbre of a musical excerpt to browse a music collection. For example the All Music Guide (http://www.allmusic.com) allows navigation by *moods* including among others: angry, mellow, sophisticated, sad, smooth, happy…

Another descriptor of interest in the context of Axmedis is the audio signal quality description scheme defined in MPEG-7 (AudioSignalQualityDS). It includes several features describing the quality of the signal. Quality information can be interesting when searching for an audio file on the Internet to decide which file should be downloaded among several search results.

Input data structures

The type of the data to be fed into the extraction algorithm should be dependent on the type of metadata to extract.

For the extraction of MPEG-7 *Low Level Descriptors* (LLDs), audio samples need to be directly provided to the extractor.

On the contrary, for *higher level descriptors*, such as rhythmic or timbre descriptors, some specific LLDs series are needed by the extractor. For example timbre modelling may use series of vectors of Mel Frequency Cepstral Coefficients as input while tempo induction may use energy in some specific frequency bands. The estimation of fingerprints is typically based on some clustering of series of some particular LLDs. Speech analysis algorithms may also take as input series of LLDs rather than audio samples.

To maximize the reuse of computed features, LLDs series extraction should be shared by all fingerprint/descriptor extractors. Rather than having a pool of algorithms working on audio samples to extract metadata, we should have a shared extractor of LLDs working on audio samples and a set of higher level extractor working on specific LLDs.

An even higher level metadata extractor such as a genre recognition algorithm would need as input the output of some other high level extractor (for example we may induce genre based on rhythmic, timbral and harmonic knowledge extracted earlier).

Output data structures: formalization in terms of MPEG-7

Most of the extracted metadata (low level or high level) may be formalized in terms of MPEG-7 and will require some complex data structure.

Here follows the description scheme of the main descriptors of interest in the context of Axmedis. Note that some lower level descriptors or some technical descriptors (in terms of music theory) may be extracted as a side product of this high-level information extraction. Yet they may be of poor interest used independently in the context of information retrieval and classification. Consequently we will focus for the moment on high-level description.

Audio classification:

Audio content may be classified according to some arbitrary taxonomy. For example, one may need to discriminate music from speech and general sounds (other than music or speech). Audio content categorised as music may be further defined in terms of musical genres while speech content may be classified in terms of voice type (male, female, child). Note that this framework may be used to classify audio content following other dimensions such as instrumentation (*guitars*, *strings*, *keyboards*, *brasses*…), mood (*aggressive*, *dark*, *dramatic*, *exotic*, *funky*, *futuristic*, *lonely*, *romantic*…) or recording type (*studio*, *live*).

MPEG-7's ClassificationScheme DS defines a set of language-independent terms that can be used for classifying some subject area. It also can organize the terms by establishing relationships amongst those terms. A ClassificationScheme DS is made up of a set of Items, each defining one term in the classification scheme. Each Item includes a unique identifier for a term (used to reference it via the term attribute), a set of human readable labels for the term, and a set of human readable definitions of what the term means.

Here is the syntax of the ClassificationScheme DS:

```
<!-- ################################################   -->
<!-- Definition of ClassificationScheme DS              -->
<!-- ################################################   -->

<complexType name="ClassificationSchemeType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence maxOccurs="1" minOccurs="1">
            <element name="Description" type="mpeg7:TextualType"
               minOccurs="0" maxOccurs="unbounded"/>
            <choice minOccurs="1" maxOccurs="unbounded">
               <element name="Item" type="mpeg7:ItemType"
                  minOccurs="1" maxOccurs="1"/>
               <element name="ItemImport"
                  type="mpeg7:ItemImportType"
                  minOccurs="1" maxOccurs="1"/>
               <element name="ClassificationSchemeRef"
                  type="mpeg7:ClassificationSchemeRefType"
                  minOccurs="1" maxOccurs="1"/>
               </choice>
         </sequence>
         <attribute name="scheme" type="mpeg7:classificationSchemeIdentifierType"
            use="required"/>
         <attribute name="mpeg7id" type="string" use="optional"/>
         <attribute name="version" type="string" use="optional"/>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of Item datatype                      -->
<!-- ################################################ -->

<complexType name="ItemType">
```

```
      <sequence minOccurs="1" maxOccurs="1">
         <element name="Label" minOccurs="0" maxOccurs="unbounded">
            <complexType>
               <simpleContent>
                  <extension base="mpeg7:TextualType">
                     <attribute name="preferred" type="boolean" use="optional"/>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
         <element name="Definition" type="mpeg7:TextualType"
            minOccurs="0" maxOccurs="unbounded"/>
         <choice minOccurs="1" maxOccurs="unbounded">
            <element name="Item" minOccurs="1" maxOccurs="1">
               <complexType>
                  <complexContent>
                     <extension base="mpeg7:ItemType">
                        <attribute name="type"  type="mpeg7:termRelationKindType"
                           use="default" value="NT"/>
                     </extension>
                  </complexContent>
               </complexType>
            </element>
            <element name="ItemImport" type="mpeg7:ItemImportType"
               minOccurs="1" maxOccurs="1"/>
            <element name="ClassificationSchemeRef"
               type="mpeg7:ClassificationSchemeRefType"
               minOccurs="1" maxOccurs="1"/>
         </choice>
      </sequence>
      <attribute name="term" type="mpeg7:controlledTermIdentifierType" use="required"/>
   </complexType>

   <!-- ################################################ -->
   <!-- Definition of ItemImport datatype                -->
   <!-- ################################################ -->

   <complexType name="ItemImportType">
      <complexContent>
         <extension base="mpeg7:ItemType">
            <attribute name="importScheme" type="QName" use="required"/>
            <attribute name="importTerm" type="string" use="required"/>
         </extension>
      </complexContent>
   </complexType>

   <!-- ################################################   -->
   <!-- Definition of ClassificationSchemeRef            -->
   <!-- ################################################   -->

   <complexType name="ClassificationSchemeRefType">
      <attribute name="schemeLocation" type="mpeg7:classificationSchemeLocatorType"
         use="optional"/>
      <attribute name="scheme" type="mpeg7:classificationSchemeIdentifierType"
         use="required"/>
   </complexType>
```

Here is the semantics of the ClassificationScheme DS:

1. Semantics of the ClassificationSchemeType:

Name	*Definition*
ClassificationSchemeType	Description scheme defining a set of terms and their relations.
Description	Indicates a human readable explanation of the classification scheme.

Name	*Definition*
Item	Describe one item in this classification scheme.
ItemImport	Describes one imported from another existing classification scheme. This allows new classification schemes to extend and build onto existing classification schemes.
ClassificationSchemeRef	References a non-external classification from which all terms are to be incorporated into this classification scheme.
Scheme	Identifies the classification with a fully qualified name of the classification scheme. The namespace URL associated with this identifier should reference the authoritative definition for the classification, if one exists.
mpeg7id	Indicates the MPEG-7 description tool(s) to which this classification scheme applies to using an XPath expression. The path is defined relative to an MPEG-7 *description*, **not** the DDL schema definition. For example: the value "//ClassificationScheme/@scheme" would indicate that the tool is appropriate for "scheme" attribute of the description tool represented by a "ClassificationScheme" element.
Version	Identifies the version of the classification scheme. The contents of this field are not specified by MPEG-7; however, the same version must always be identified with the same string.

2. Semantics of the ItemType:

Name	*Definition*
ItemType	Datatype representing a single term definition in the ClassificationScheme DS.
Label	Indicates the human readable label for the item. It is possible to have multiple labels for the same item, possibly in different languages. The languages of a label is indicated by the xml:lang attribute.
preferred	Indicates whether or not this is the preferred label for this term. In the case where multiple labels exist, only one label per language shall be marked preferred.
Definition	Indicates the human readable explanation of the item. It is possible to have multiple definitions in different languages for the same item. The languages of a description is indicated by the xml:lang attribute.
Item	Describes a set of items related to the containing item.
Type	Indicates the type of relation existing between the contained item and the containing item. By default, it is "NT", indicating that the contained item is narrower in meaning than this item.
ClassificationSchemeRef	A reference to a ClassificationScheme that is inserted at the current level of the hierarchy. See below for detailed explanation of including classification schemes via referencing.
Term	The unique identifier for this term in the classification scheme. All items within a single classification must be unique.

3. Semantics of the ItemImportType:

Name	*Definition*
ItemImportType	Datatype describing a term imported from an existing classification scheme.
Term	Identifies the term in the current classification scheme. It may be different from the imported term's original identifier, which is designated by importTerm.
importScheme	The identifier of the classification scheme from which the term is being imported.
importTerm	Identifies the term identifier in the classification scheme from which is being imported.

4. Semantics of the ClassificationSchemeRefType:

Name	*Definition*
ClassificationSchemeRefTyp e	A reference to the ClassificationScheme DS.
Scheme	Identifies the classification scheme being referenced.
schemeLocation	References the location of the definition of the referenced classification scheme.

One classification may be imported into another using ClassificationSchemeRef. This is useful when one wants to combine several independent classification schemes into a single larger classification scheme.

Let ReferencedCS refer to the classification scheme referenced by the definition of classification scheme DefineCS. An item that is not contained within another item definition is called a top-level item. Then the reference to a classification scheme within the definition of another classification scheme is interpreted as follows.

1. The set of items defined in ReferencedCS is added to the set of items in DefineCS. It is an error if an item with the same value for term occurs in both ReferencedCS and DefineCS.

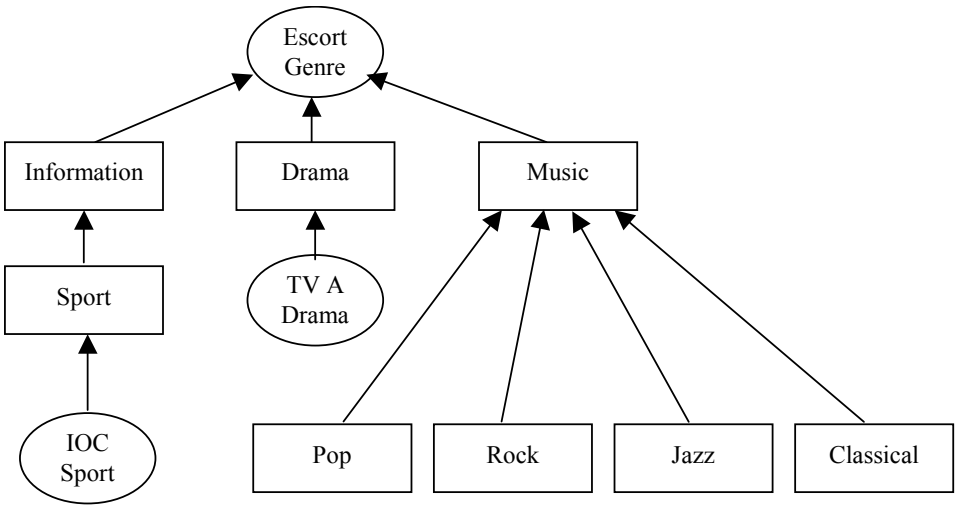
2. The value of term, and the values the labels and definitions are incorporated unmodified into DefineCS from ReferencedCS.

3. The set of term relations defined in ReferncedCS is added to the set of term relations in DefineCS.

4. If the ClassificationSchemeRef is at the topmost level of DefineCS, then the top-level elements in ReferencedCS are added to the set of top-level item in DefineCS.

5. If the ClassificationSchemeRef is not at the topmost level - i.e. occurs within an item definition - then all top-level elements of ReferencedDS are related to the containing term by a "narrower term" relation.

The following figure shows an example of a simple classification for genre identified as "Escore:Genre2.4". In the figure round boxes represent classification schemes and square boxes items.



The "Escort:Genre2.4" Genre Classification Scheme.

In this example, there are three items at the highest level: information, drama, and music. Under the information category there a more detailed term: sports. Rather than defining a complete classification for

sports, this example shows how an existing classification can be "spliced" into the classification scheme hierarchy. In this case the existing "IOC Sports" classification scheme is added under the Sports item. Similarly, the drama category includes the "TVE Drama" classification scheme. For music, the sub-items (narrower in meaning than their containing item) are "pop", "rock", "jazz", and "classical".

```
<ClassificationScheme
      scheme="Escort2_4:Content"
      mpg7id="CreationInformation/Classification/Genre">

      <Item term="1">
            <Label xml:lang="en">Information</Label>
            <Definition xml:lang="en">Generic news</Definition>

            <Item term="1.1">
                  <Label xml:lang="en">Sport</Label>
                  <Definition xml:lang="en">Sports news</Definition>

                  <ClassificationSchemeRef
                        xmlns:IOC="http://www.ioc.org"
                        scheme="IOC:Sports">
            </Item>
      </Item>

      <Item term="2">
            <Label xml:lang="en">Drama</Label>
            <Definition xml:lang="en">Dramatic Programs</Definition>

            <ClassificationSchemeRef
                  xmlns:TVE = "http://www.tvid.org"
                  scheme="TVE:Drama"/>
      </Item>

      <Item term="3">
            <Label xml:lang="en">Music</Label>
            <Definition xml:lang="en">Musical Programs</Definition>
            <Item term="3.1">
                  <Label xml:lang="en">Rock</Label>
            </Item>
            <Item term="3.2">
                  <Label xml:lang="en">Pop</Label>
            </Item>
            <Item term="3.3">
                  <Label xml:lang="en">Jazz</Label>
            </Item>
            <Item term="3.4">
                  <Label xml:lang="en">Classical</Label>
            </Item>
      </Item>
</ClassificationScheme>
```

Content navigation and access: structure of audio content and summarization

This section provides MPEG-7 description schemes helping in browsing and navigating into the audio content. It provides schemes to store audio segments representative of the structure of the content. For example, the typical structure of a pop song may look like: intro verse, chorus, second verse, chorus, bridge, third verse, chorus, coda, outtro. In the case of spoken content, one may wish to structure the audio content in terms of speakers or subject.

The HierarchicalSummary DS is constructed around the generic notion of temporal segments of AV data, described by HighlightSegments. Each HighlightSegment contains locators to the AV data, to provide access to the associated key-videoclip or key-audioclip, to key-frames and to key-sounds and may also contain textual annotation referring to key-themes. These audiovisual segments are grouped into summaries, or highlights, using the HighlightSummary description scheme. Such summaries may correspond to two different themes and could provide alternative views on the original AV content. The HighlightSummary description scheme is recursive in nature, enabling summaries to contain other summaries. This capability can be used to build a variety of hierarchical summaries, i.e. to describe content at different granularities. Additionally, multiple summaries may be grouped together using the HierarchicalSummary description scheme.

Here is the syntax of the description scheme involved in the summarization definition:

```
<!-- ################################################ -->
<!-- Definition of Summarization DS                   -->
<!-- ################################################ -->

<complexType name="SummarizationType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="Summary" type="mpeg7:SummaryType"
               minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of Summary DS                         -->
<!-- ################################################ -->

<complexType name="SummaryType" abstract="true">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="Name" type="mpeg7:TextualType"
               minOccurs="0" maxOccurs="1"/>
            <element name="SourceLocator" type="mpeg7:MediaLocatorType"
               minOccurs="0" maxOccurs="1"/>
            <element name="SourceInformation" type="mpeg7:ReferenceType"
               minOccurs="0" maxOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of HierarchicalSummary DS             -->
<!-- ################################################ -->

<complexType name="HierarchicalSummaryType">
   <complexContent>
      <extension base="mpeg7:SummaryType">
         <sequence>
            <element name="SummaryThemeList" type="mpeg7:SummaryThemeListType"
               minOccurs="0" maxOccurs="1"/>
            <element name="HighlightSummary" type="mpeg7:HighlightSummaryType"
               minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="components" use="required">
            <simpleType>
               <list>
```

```
                  <simpleType>
                     <restriction base="string">
                        <enumeration value="keyVideoClips"/>
                        <enumeration value="keyAudioClips"/>
                        <enumeration value="keyFrames"/>
                        <enumeration value="keySounds"/>
                        <enumeration value="keyThemes"/>
                     </restriction>
                  </simpleType>
               </list>
            </simpleType>
         </attribute>
         <attribute name="hierarchy" use="required">
            <simpleType>
               <restriction base="string">
                  <enumeration value="independent"/>
                  <enumeration value="dependent"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of SummaryThemeList DS                -->
<!-- ################################################ -->

<complexType name="SummaryThemeListType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="SummaryTheme" minOccurs="1" maxOccurs="unbounded">
               <complexType>
                  <simpleContent>
                     <extension base="mpeg7:TextualType">
                        <attribute name="id" type="ID"
                           use="required"/>
                        <attribute name="parentId" type="IDREF"
                           use="optional"/>
                     </extension>
                  </simpleContent>
               </complexType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of HighlightSummary DS                -->
<!-- ################################################ -->

<complexType name="HighlightSummaryType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="Name" type="mpeg7:TextualType"
               minOccurs="0" maxOccurs="1"/>
            <element name="HighlightSegment" type="mpeg7:HighlightSegmentType"
               minOccurs="1" maxOccurs="unbounded"/>
            <element name="HighlightChild" type="mpeg7:HighlightSummaryType"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="level" type="integer"
            use="optional"/>
         <attribute name="duration" type="mpeg7:mediaDurationType"
```

```
            use="optional"/>
         <attribute name="numKeyFrames" type="nonNegativeInteger"
            use="optional"/>
         <attribute name="fidelity" type="mpeg7:zeroToOneType"
            use="optional"/>
         <attribute name="themeIds" type="IDREFS"
            use="optional"/>
      </extension>
   </complexContent>
</complexType>

<!-- ################################################ -->
<!-- Definition of HighlightSegment DS                -->
<!-- ################################################ -->

<complexType name="HighlightSegmentType">
   <complexContent>
      <extension base="mpeg7:DSType">
         <sequence>
            <element name="Name" type="mpeg7:TextualType"
               minOccurs="0" maxOccurs="1"/>
            <element name="KeyVideoClip" type="mpeg7:VideoSegmentLocatorType"
               minOccurs="0" maxOccurs="1"/>
            <element name="KeyAudioClip" type="mpeg7:AudioSegmentLocatorType"
               minOccurs="0" maxOccurs="1"/>
            <element name="KeyFrame" type="mpeg7:ImageLocatorType"
               minOccurs="0" maxOccurs="unbounded"/>
            <element name="KeySound" type="mpeg7:AudioSegmentLocatorType"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="themeIds" type="IDREFS" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

Here is the semantics of the different terms involved in the summarization definition:

Semantics of the SummarizationType:

Name	*Definition*
SummarizationType	Specifies a set of Summary elements.
Summary	An AV summary of AV content or a related group of summaries. See section **Errore. L'origine riferimento non è stata trovata.**.

Semantics of the SummaryType:

Name	*Definition*
SummaryType	An abstract DS from which the following description schemes are derived: • HierarchicalSummary DS • SequentialSummary DS
Name	Specifies the name of an instantiation of the Summary DS.
SourceLocator	Specifies location of the original (source) AV content that is summarized.
SourceInformation	References an element of a description of the original (source) AV content. Shall refer to a valid id attribute of a description element.

Semantics of the HierarchicalSummaryType:

Name	*Definition*
HierarchicalSummaryType	Specifies a group of summaries that contain hierarchically ordered audio-visual segments. A HierarchicalSummary element contains HighlightSummary elements, each of which specify a single, complete summary.
SummaryThemeList	Specifies a list of textual themes associated with parts of the HierarchicalSummary.
HighlightSummary	Specifies a single AV summary, which can consist of a hierarchy of components. Each HighlightSummary represents an alternative view on the AV content. See section **Errore. L'origine riferimento non è stata trovata.**.
components	Specifies a list of the types of summary components included in a HierarchicalSummary. The types allowed are defined as follows. • *keyVideoClips* - The summaries shall contain key-videoclips, possibly ordered hierarchically. Such video clips form a video summary of a particular duration. A key-videoclip can be a video segment from the content, or from related media. • *keyAudioClips* - The summaries shall contain key-audioclips, possibly ordered hierarchically. Such audio clips form an audio summary of a particular duration. A key-audioclip can be an audio segment from the content, or from related media. • *keyFrames* - The summaries shall contain key-frames, possibly ordered hierarchically. A summary may contain a higher number of key-frames on each subsequent level of its hierarchy, to provide different levels of detail. A key-frame can be a specific frame from a video segment, or an image that is not in the video, possibly a synthetic image (pre-composed from multiple images). • *keySounds* - The summaries shall contain key-sounds, possibly ordered hierarchically. A summary may contain a higher number of key-sounds on each subsequent level of its hierarchy, to provide different levels of detail. A key-sound may correspond to key words in speech, sound effects, emotional sounds, exploding sounds, specific instrument sounds, and possibly synthetic sounds. • *keyThemes* - The summaries shall contain videoclips and/or audioclips, possibly ordered hierarchically, as well as textual descriptions of associated events or themes. Each summary is a collection of videoclips and/or audioclips referring to particular key-events or themes. Key-events or themes may be described textually by key-words.
hierarchy	Indicates the type of the hierarchy with respect to the parent-child relationships between elements at different levels of the hierarchy. This attribute may be used to eliminate unnecessary duplication of information in a hierarchy of elements. The types of the hierarchy are defined as follows. • *independent* - The information in the elements on a single level of a hierarchy completely specifies a particular summary, without reference to the information in the parent elements of these elements. Information in the parent elements shall not be re-used in the children elements. • *dependent* - The information in children elements in a hierarchy adds to, or refines, the information in the parent elements. Information in the parent elements shall be re-used in the children elements.

Name	*Definition*
	Note that the value of this attribute may be ignored if none of the HighlightSummary elements of a HierarchicalSummary contain HighlightChild elements.

Semantics of the SummaryThemeListType:

Name	*Definition*
SummaryThemeListType	Defines a list of SummaryTheme elements.
SummaryTheme	Describes an event or theme in textual form, in terms of which a video can be summarized.
id	Identifies an instantiation of a SummaryTheme element.
parentId	Refers to another SummaryTheme element that corresponds to the parent- or super-theme in a conceptual hierarchy of themes (optional). Shall refer to the valid id attribute of a SummaryTheme element.

Semantics of the HighlightSummaryType:

Name	*Definition*
HighlightSummaryType	Specifies a single summary or part of a summary. Contains a set of audio-visual segments that form a summary. A HighlightSummary element may contain HighlightSummary elements as its children, in which case a tree-based hierarchy of summary elements may be formed. Each tree has a single root element that is part of a HierarchicalSummary element and all elements in a single tree correspond to the same summary (at different levels of detail).
Name	Name of a particular HighlightSummary element.
HighlightSegment	Describes an audio-visual segment by its key-videoclip and/or key-audioclip, its key-frames, key-sounds and key-themes. See section **Errore. L'origine riferimento non è stata trovata.**.
HighlightChild	Describes a child HighlightSummary element that describes the current summary in more detail. Child HighlightSummary elements are used to form a tree-based hierarchy of summary components. A summary at a particular level of detail is to be constructed by combining information from all HighlightChild nodes at or up to the same level in a single tree. If the hierarchyType of the hierarchical summary tree is "independent", the summary components (key-videoclips, key-audioclips, key-frames and key-sounds) in all children HighlightSummary elements at the same level of a single tree shall be combined to define a single AV summary. If the hierarchyType is "dependent", the summary components in all children HighlightSummary elements up to a particular level shall be added to the components of their parent tree nodes (recursively up the tree) to define a single, complete AV summary. In the latter case, all elements up to a particular level in the tree contribute to an AV summary at a particular level of detail.
level	Indicates the level of a HighlightSummary element in a hierarchy (optional). The root HighlightSummary element in a hierarchy has level 0, its children HighlightSummary elements have level 1, etc.
duration	Indicates the temporal duration of the HighlightSegments contained in a HighlightSummary element (optional). Indicates the total duration of key-videoclips in a video summary; indicates the total duration of key-audioclips in an audio summary.
numKeyFrames	Indicates the total number of key-frames contained in the set of HighlightSegment elements in a HighlightSummary element (optional).
fidelity	Indicates how well the information in the HighlightSummary element is represented by the information in its parent HighlightSummary element, on a numerical scale between 0.0 and 1.0 (optional). Values closer to 1.0 correspond to better representations of this

Name	*Definition*
	element by the associated parent element, while values closer to 0.0 correspond to worse representations.
themeIds	A list of references to SummaryTheme identifiers indicating key-themes (key-events) common to all children HighlightSummary and HighlightSegment elements (optional). Shall refer to valid id attributes of SummaryTheme elements.

Semantics of the HighlightSegmentType:

Name	*Definition*
HighlightSegmentType	Specifies an audio-visual segment. May contain a video segment (key-videoclip), an audio segment (key-audioclip), images (key-frame) or sounds (key-sound).
Name	Identifies the segment by name.
KeyVideoClip	Specifies the location of a key-videoclip. A key-videoclip is an (audio-)visual segment of AV content, which can be used for navigation, browsing and summarization. See section **Errore. L'origine riferimento non è stata trovata.** for the definition of VideoSegmentLocator.
KeyAudioClip	Specifies the location of a key-audioclip. A key-audioclip is an audio segment of AV content, which can be used for navigation, browsing and summarization. See section **Errore. L'origine riferimento non è stata trovata.** for the definition of AudioSegmentLocator.
KeyFrame	Specifies the location of a key-frame. A key-frame is a single video frame in AV content, which can be used for navigation, browsing and summarization. See section **Errore. L'origine riferimento non è stata trovata.** for the definition of ImageLocator.
KeySound	Specifies the location of a key-sound. A key-sound is a single sound in AV content, which can be used for navigation, browsing and summarization. See section **Errore. L'origine riferimento non è stata trovata.** for the definition of AudioSegmentLocator.
themeIds	A list of references to SummaryTheme identifiers indicating key-themes (key-events) common to all children HighlightSummary and HighlightSegment elements (optional). Shall refer to valid id attributes of SummaryTheme elements.

2.5.3 Descriptor/Metadata Extractors from Video Files (FHGIGD …..)

The implementation of metadata extractors from video files within AXMEDIS is limited to integration of one or more available content description algorithms for video. On the one hand the selection depends on the requirements within AXMEDIS, the currently used descriptors, and the availability/accessibility existing libraries. On the other hand the availability of the algorithms determines the meta data extraction algorithms for video. Depending on the available algorithms a suitable scheme from MPEG-7 will be chosen or extended.

Basic output data structures

In MPEG-7 visual descriptors are defined, which are primarily considered for the meta data extracted from videos. As videos consists of a (time) series of images the basic data type is “VisualTimeSeriesType”.

```
<complexType name="VisualTimeSeriesType" abstract="true">
   <sequence>
      <element name="TimeIncr" type="mpeg7:mediaDurationType"/>
   </sequence>
   <attribute name="offset" type="mpeg7:mediaDurationType"
         use="defaultoptional" valuedefault="PT0S"/>
</complexType>
```

In MPEG-7, regular and irregular time series are distinguished depending on the (non-) constant interval size of the collected descriptors.

Regular Visual Time Series

```
<complexType name="RegularVisualTimeSeriesType" final="#all">
   <complexContent>
      <extension base="mpeg7:VisualTimeSeriesType">
         <sequence>
            <element name="Descriptors" type="mpeg7:VisualDType"
                  minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Semantics of the RegularVisualTimeSeries:

DescriptorID	This elementfield, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier specifies specifies the descriptor type accommodated in the time series.
NumOfDescriptorsNum	This elementfield, which is only present in the binary representation, specifies the number of descriptor instances accommodated in the time series.
IsRandomAccess	This elementfield, which is only present in the binary representation, specifies the access mode, which is either: • random access if the flag is set to 1; in this case DescriptorLength and BitStuffing elements are present in the binary representation • no random access if the flag is set to 0; in this case no bit stuffing is allowed and descriptor instances are not padded, which means they may have different lengths
DescriptorLength	This field, which is only present in the binary representation, specifies the length of each descriptor instance in bytes. The value of this element is the size of the largest descriptor instance, aligned to a byte boundary by bit stuffing using 0-7 '1' bits.
TimeIncr	This element specifies the default time interval. The time interval is defined as an interval between descriptor locations. An interval that follows a descriptor is associated with the descriptor. The type of this element "mediaDurationType" is specified in ISO/IEC 15938-5.
IsOffset	This field, which is only present in the binary representation, signals the presence of the offset attribute. If it is equal to 1 (true) offset is present, if 0 (false) offset is not specified (i.e. default value should be used).
offset	This attribute specifies the offset, i.e., the interval between the starting time point of a given time span and the location of the first descriptor. The default value is zero (represented as "PT0S" in DDL). This attribute is illustrated as "Offset" in **Errore. L'origine riferimento non è stata trovata.**.
BitStuffing	This field, which is only present in the binary representation, specifies stuffing bits (Aa sequence of '1's) stuffing bits to align on the byte boundary. to align the descriptor to a byte boundary.

Descriptors	This element contains the instantiation of specifies the visual descriptor accommodated in this time series. Only one type of child descriptor is allowed to be instantiated. Its binary syntax and semantics follow those of the assigned descriptor. In random access mode, if the size of a particular descriptor instance is smaller than DescriptorLength, it is padded with the required number of '1' bits.

Irregular Visual Time Series

```
<complexType name="IrregularVisualTimeSeriesType" final="#all">
   <complexContent>
      <extension base="mpeg7:VisualTimeSeriesType">
         <sequence minOccurs="1" maxOccurs="unbounded">
            <element name="Descriptors" type="mpeg7:VisualDType"/>
            <element name="Interval" type="mpeg7:unsigned32"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Semantics of IrregularVisualTimeSeries:

DescriptorID	This elementfield, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier specifies the descriptor type accommodated in the time series.
NumOfDescriptorsNum	This elementfield, which is only present in the binary representation, specifies the number of descriptor instances accommodated in the time series.
IsRandomAccess	This elementfield, which is only present in the binary representation, specifies the access mode, which is either: • random access if the flag is set to 1; in this case DescriptorLength and BitStuffing elements are present in the binary representation • no random access if the flag is set to 0; in this case no bit stuffing is allowed and descriptor instances are not padded, which means they may have different lengths
DescriptorLength	This elementfield, which is only present in the binary representation, specifies the length of each descriptor instance in bytes. The value of this element is the size of the largest descriptor instance, aligned to a byte boundary by bit stuffing using 0-7 '1' bits.
IsShortInterval	This elementfield, which is only present in the binary representation, indicates the size of the ShortInterval/LongInterval field. 1 (true) for 8-bit unsigned integer("unsigned8") while 0 (false) for 32-bit unsigned integer("unsigned32"). If IsShortInterval is set to 1, then 8-bit unsigned integer ("unsigned8") is used. If IsShortInterval is set to 0, then a 32-bit unsigned integer ("unsigned32") is used.
TimeIncr	This element specifies the base unit of the time interval. The time interval between descriptor locations is specified as a multiple of this base unit. The type of this element, MediaDurationType, is specified in ISO/IEC 15938-5.
IsOffset	This elementfield, which is only present in the binary representation, signals the presence of the offset attribute. If it is equal to 1 (true) offset is present, if 0 (false) offset is not specified (i.e. default value should be used).If IsOffset is set to 1 then the offset attribute is present. If IsOffset is set to 0 then the offset attribute is not specified (i.e. the default value should be used).

offset	This attribute specifies the offset, i.e., the interval between the starting time point of a given time span and the location of the first descriptor. The default value is zero (represented as "PT0S" in DDL). This element is illustrated as "Offset" in **Errore. L'origine riferimento non è stata trovata.**.
BitStuffing	This field, which is only present in the binary representation, specifies stuffing bits (a sequence of '1's) to align the descriptor to a byte boundary.
Descriptors	This element contains the instantiation of specifies the visual descriptor accommodated in this time series. Only one type of child descriptor is allowed to be instantiated. Its binary syntax and semantics follow those of the assigned descriptor. In random access, if the size of a particular descriptor instance is smaller than DescriptorLength, it is padded with the required number of '1' bits.
Interval/ShortInterval/LongInterval	This element specifies the time interval between the current and the preceding descriptor. The value of the element is specified in units defined by TimeIncr.

Within MPEG-7 feature descriptors are already defined including

- color,
- texture,
- shape, and
- motion.

MPEG-7 Colour descriptors

For the description of colour MPEG-7 includes several descriptors like:

- **ColorSpace** is a supporting tool to express in which colour space the colour descriptors are expressed.
- **ColorQuantization** is also a supporting and provides a mapping from the floating point values to an integer representation.
- **DominantColor** specifies a set of dominant colours and targets content based retrieval for colors.
- **ScalableColor** specifies a colour distribution .
- **ColorLayout** specifies a global spatial colour distribution .
- **ColorStructure** specifies a local spatial colour distribution .

MPEG-7 Texture descriptors

Texture so far are described in MPEG-7 by:

- **HomogeneousTexture** describes region texture by a frequency specific energy and energy deviation.
- **TextureBrowsing** specifies perceptual characterization of a texture (like regularity, coarseness, and directionality).
- **EdgeHistogram** specifies the spatial distribution of edges in local regions (sub-images).

MPEG-7 Shape descriptors

The already defined shape descriptor in MPEG are:

- **RegionShape** specifies a region-based shape of an object.
- **ContourShape** specifies a closed contour of a 2D object or region.
- **Shape3D** specifies the intrinsic shape description for 3D mesh models.

Motion

Different kinds of motion descriptors are already defined in MPEG-7:

- **CameraMotion** specifies 3D camera motion parameters.
- **MotionTrajectory** specifies the motion trajectory of a moving object.
- **ParametricMotion** specifies the motion of objects in video sequences.
- **MotionActivity** captures the notion of "intensity of motion" in a video segment (intensity of activity, direction of activity, spatial distribution of activity, spatial localization of activity, and temporal distribution of activity).

Module Profile		
Descriptor Extractor Prototype for Video Files		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker (FHGIGD)	
Responsible Partner	FHGIGD (fingerprinting)	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
None	None	None
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	Will be identified during the project.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Will be identified during the project.	Will be identified during the project.	Will be identified during the project.

2.5.4 Descriptor/Metadata Extractors for General Digital Resources (FHGIGD, EPFL…..)

Only limited metadata information can be extracted from general digital resources as these files are only considered as binary files. Typical information, which might be relevant but cannot extracted from the digital resource directly include:

- Resource name
- Creation date
- Last modification date

In contrast to the previous information the only information that can be extracted automatically is

- resource size,
- cryptographic hash, and
- bit value distribution related information.

However, only the resource size and the corresponding cryptographic hash function are reasonable.

In MPEG-7 this kind of meta-information (for verification of the digital resource) is not consider so far. Ideally, each description should contain a cryptographic hash function to ensure the link between the object and meta-data. This is not yet foreseen within MPEG-7:

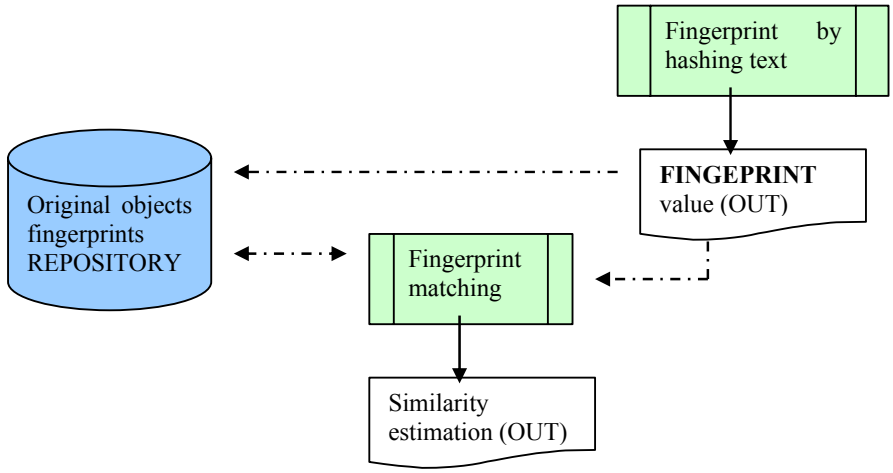
```
<!-- ################################################ -->
<!--  Definition of Basic Description Type                    -->
<!-- ################################################ -->

<complexType name="BasicDescriptionType" abstract="true">
   <sequence>
      <element name="Relationships" type="mpeg7:GraphType"
         minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

Within AXMEDIS it has to be identified, how cryptographic hash values can be integrated best into MPEG-7 and the relationship with MPEG-21.

Module Profile		
Descriptor Extractor Prototype for General Digital Resources		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker (FHGIGD)	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
None	None	None

DE3.1.2D – Framework

User Interface
None
Used Libraries
BeeCrypt (?)
Botan (?)
CryptLib (?)



In MPEG-7 this kind of meta-information (for verification of the text documents) is not consider so far. Within AXMEDIS it has to be identified, how cryptographic hash values can be integrated best into MPEG-7 and the relationship with MPEG-21.

Module Profile		
Fingerprint Extractors for Documents		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	Native language necessary: C++	
Responsible Name	Marco Fabbri	
Responsible Partner	DIPITA	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, GNU/Linux	
Interfaces with other tools:	Name of the communicating tools	Name of the communicating tools
File Formats Used	Shared with	Shared with
User Interface	Development model, language, etc.	Development model, language, etc.
	Native language with a well-defined language independent interface, e.g. CORBA	Native language with a well-defined language independent interface, e.g. CORBA
	Library	Library
	Multithreaded	Multithreaded
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
BeeCrypt (?)		LGPL
Botan (?)		BSD-style
CryptLib (?)		GPL and standard commercial

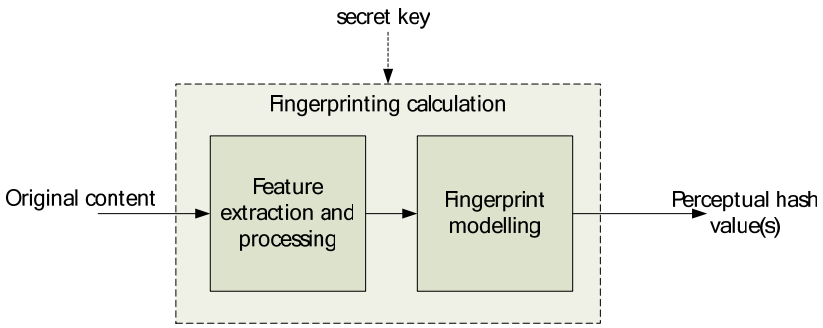
		license

2.6.2 Fingerprint Extractors for Audio Files (FHGIGD)

The Fingerprint extractors for audio files automatically calculate a digest describing its main characteristics in way suitable for automatic verification of AXMEDIS objects. Thus the descriptor is a low level description according to the previous definition.

Audio files are streams. These input stream is segmented. That means that for each segment a "sub-fingerprint" is calculated. Depending on the length of the input sequence, typically a request to the database do not result in a single fingerprint but in an array of fingerprints.

The characteristic processing steps are shown in the following figure:



- **Feature extraction and processing:** The input signal is pre-processed, which depends on the data type. For audio typical pre-processing operations are down-sampling, format conversion, and band-pass filtering. In the case of audio or video the input data are segmented and so-called "sub-fingerprints" are calculated. Features are generally extracted from a transformation domain. This transformation domain redundancy is decreased (similar to compression). Within this transformation domain relevant features are extracted. In a post-processing specific relative measures can be derived.

- **Fingerprint modelling:** The multi-dimensional input vector sequence is mapped to a single vector to produce compact fingerprints. This can also include a binarisation.

Output data structures: formalization in terms of MPEG-7

Within MPEG-7 low-level descriptors for audio are already defined. Below the main descriptors of MPEG-7 for the fingerprinting of audio file are described:

```
<!-- ################################################################### -->
<!-- Definition of AudioLLDScalarType                                    -->
<!-- ################################################################### -->
<complexType name="AudioLLDScalarType" abstract="true">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <choice>
            <element name="Scalar" type="float"/>
            <element name="SeriesOfScalar">
               <complexType>
```

```
                <complexContent>
                   <extension base="mpeg7:SeriesOfScalarType">
                      <attribute      name="hopSize"      type="mpeg7:mediaDurationType"
use="optional" default="10F1000"/>
                   </extension>
                </complexContent>
             </complexType>
          </element>
       </choice>
    </extension>
  </complexContent>
</complexType>
```

Name	Definition
AudioLLDScalarType	Abstract definition inherited by all scalar datatype audio descriptors.
Scalar	Value of the descriptor
SeriesOfScalar	Scalar values for sampled-series description of an audio segment. Use of this scalable series datatype promotes compatibility between sampled descriptions.
hopSize	Time interval between data samples for series description. The default value is PT10N1000F which is 10 milliseconds. Values other than the default shall be integer multiples/divisors of 10 milliseconds. This will ensure compatibility of descriptors sampled at different rates.

```
<!-- ################################################################### -->
<!-- Definition of AudioLLDVectorType                                    -->
<!-- ################################################################### -->
<complexType name="AudioLLDVectorType" abstract="true">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <choice>
        <element name="Vector" type="mpeg7:floatVector"/>
        <element name="SeriesOfVector">
          <complexType>
            <complexContent>
              <extension base="mpeg7:SeriesOfVectorType">
                <attribute      name="hopSize"      type="mpeg7:mediaDurationType"
use="optional" default="10F1000"/>
              </extension>
            </complexContent>
          </complexType>
        </element>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

Name	Definition
AudioLLDVectorType	Abstract definition inherited by all vector datatype audio descriptors.
Vector	Vector value of descriptor
SeriesOfVector	Vector values for sampled-series description of an audio segment. Use of this scalable series datatype promotes compatibility between sampled descriptions.
hopSize	Time interval between data samples for series description. The default value is PT10N1000F which is 10 milliseconds. Values other than the default shall be integer multiples/divisors of 10 milliseconds. This will ensure compatibility of descriptors sampled at different rates.

Module Profile		
Fingerprint Extractor Prototype for Audio Files		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker (FHGIGD)	
Responsible Partner	FHGIGD (low level descriptors)	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
RtAudio	RtAudio V3.0	BSD-style open source
PortAudio	PortAudio V18	BSD-style open source
Libsndfile (?)	1.0.11	LGPL
FFTW	V3.0.1	GPL and Non-free license (see http://web.mit.edu/tlo/www/)

2.6.3 Fingerprint Extractors for Video Files (FHGIGD, EPFL…..)

Video files are related to audio as video also is time dependent: They are streams. The input stream is segmented. For each segment a “sub-fingerprint” is calculated. Depending on the length of the input sequence, typically a request to the database do not result in a single fingerprint but in an array of fingerprints.

The characteristic processing steps the same as for audio

- **Feature extraction and processing:** The input signal is pre-processed, which depends on the data type. Typical pre-processing operations includes resizing or colour conversion. The input data are segmented and so-called "sub-fingerprints" are calculated. Features are generally extracted from a transformation domain. This transformation domain redundancy is decreased (similar to compression). Within this transformation domain relevant features are extracted. In a post-processing specific relative measures can be derived.

- **Fingerprint modelling:** The multi-dimensional input vector sequence is mapped to a single vector to produce compact fingerprints. This can also include a binarisation.

Output data structures: formalization in terms of MPEG-7

As described before, MPEG-7 defines data type for regular and irregular visual time series depending on the (non-) constant intervals between succeeding descriptors.

```
<!-- ################################################ -->
<!--   Definition of the TimeSeriesDatatype        -->
<!-- ################################################ -->
<complexType name="TimeSeriesType" abstract="true">
   <sequence>
      <element name="TimeIncr" type="mpeg7:mediaDurationType"/>
   </sequence>
   <attribute      name="offset"      type="mpeg7:mediaDurationType"      use="optional"
default="PT0S"/>
</complexType>

<!-- ################################################ -->
<!--   Definition of the VisualTimeSeriesDatatype        -->
<!-- ################################################ -->
<complexType name="VisualTimeSeriesType" abstract="true">
   <sequence>
      <element name="TimeIncr" type="mpeg7:mediaDurationType"/>
   </sequence>
   <attribute      name="offset"      type="mpeg7:mediaDurationType"      use="optional"
default="PT0S"/>
</complexType>

<!--   Definition of the RegularTimeSeries Datatype        -->
<complexType name="RegularTimeSeriesType" final="#all">
   <complexContent>
      <extension base="mpeg7:VisualTimeSeriesType">
         <sequence>
            <element     name="Descriptor"     type="mpeg7:VisualDType"     minOccurs="1"
maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<!--   Definition of the IrregularTimeSeries Datatype        -->
<complexType name="IrregularTimeSeriesType" final="#all">
   <complexContent>
      <extension base="mpeg7:VisualTimeSeriesType">
         <sequence minOccurs="1" maxOccurs="unbounded">
            <element name="Descriptor" type="mpeg7:VisualDType"/>
            <element name="Interval" type="mpeg7:unsigned32"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

As no low level descriptor data type is defined for video data, which can store video fingerprints, a descriptor has to be defined within AXMEDIS. This is related to the existing ColorStructureType:

```
<complexType name=”ColorStructureType” final=”#all”>
   <complexContent>
      <extension base=”mpeg7:VisualDType”>
         <sequence>
            <element name=”Values”>
               <simpleType>
                  <restriction>
```

```
                  <simpleType>
                     <list itemType=”mpeg7:unsigned8”/>
                  </simpleType>
                  <minLength value=”1”/>
                  <maxLength value=”256”/>
               </restriction>
            </simpleType>
         </element>
      </sequence>
      <attribute name=”colorQuant” type=”mpeg7:unsigned3” use=”required”/>
   </extension>
</complexContent>
</complexType>
```

However, instead of describing the colour structure a “VideoLLDScalar” and a “VideoLLDVectorType” are proposed, which have to be further evaluated according to the needs within AXMEDIS and general needs:

```
<!-- ################################################################### -->
<!-- Definition of VideoLLDScalarType                                    -->
<!-- ################################################################### -->
<complexType name="VideoLLDScalarType" abstract="true">
   <complexContent>
      <extension base="mpeg7:VisualDType">
         <choice>
            <element name="Scalar" type="float"/>
            <element name="SeriesOfScalar">
               <complexType>
                  <complexContent>
                     <extension base="mpeg7:SeriesOfScalarType">
                        <attribute      name="hopSize"      type="mpeg7:mediaDurationType"
use="optional" default="10F1000"/>
                     </extension>
                  </complexContent>
               </complexType>
            </element>
         </choice>
      </extension>
   </complexContent>
</complexType>


<!-- ################################################################### -->
<!-- Definition of VideoLLDVectorType                                    -->
<!-- ################################################################### -->
<complexType name="VideoLLDVectorType" abstract="true">
   <complexContent>
      <extension base="mpeg7: VisualDType ">
         <choice>
            <element name="Vector" type="mpeg7:floatVector"/>
            <element name="SeriesOfVector">
               <complexType>
                  <complexContent>
                     <extension base="mpeg7:SeriesOfVectorType">
                        <attribute      name="hopSize"      type="mpeg7:mediaDurationType"
use="optional" default="10F1000"/>
                     </extension>
                  </complexContent>
               </complexType>
            </element>
         </choice>
      </extension>
   </complexContent>
</complexType>
```

Module Profile		
Fingerprint Extractor Prototype for Video Files		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	Native language necessary: C++	
Responsible Name	Martin Schmucker (FHGIGD)	
Responsible Partner	FHGIGD (fingerprinting)	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
FFTW (?)	V3.0.1	GPL and Non-free license (see http://web.mit.edu/tlo/www/)
FFMPEG (?)		LGPL
FOBS (?)		LGPL

2.6.4 Fingerprint Extractors for Any Digital Files (FHGIGD, EPFI…..)

The fingerprint extractors for any (arbitrary) digital file cannot derive characteristics of the content. Thus only the cryptographic hash function can be calculated to verify the AXMEDIS objects' identity.

Module Profile		
Descriptor Extractor Prototype for General Digital Resources		
Executable or Library(Support)	Library	
Single Thread or Multithread	Multithreaded	
Language of Development	C++	
Responsible Name	Martin Schmucker (FHGIGD)	
Responsible Partner	FHGIGD	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)

File Formats Used	Shared with	File format name or reference to a section
None	None	None
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
BeeCrypt (?)		LGPL
Botan (?)		BSD-style
CryptLib (?)		GPL and standard commercial license

2.6.5 Fingerprint Extractors for metadata (FHGIGD, EPFl…..)

For the verification of the objects metadata only cryptographic hash functions are feasible. This hash function is applied to all kinds of meta-data of an AXMEDIS object.

In MPEG-7 this kind of meta-information (for verification of the meta-data) is not consider so far. Ideally, each description should contain a cryptographic hash function to ensure the link between the object and meta-data. This is not yet foreseen within MPEG-7:

```
<!-- ################################################ -->
<!--  Definition of Mpeg7RootType                     -->
<!-- ################################################ -->

<complexType name="Mpeg7RootType" abstract="true">
   <attribute ref="xml:lang" use="optional"/>
   <attribute name="name" type="NCName" use="optional"/>
   <attribute name="version" type="string" use="optional"/>
   <attribute name="creationTime" type="mpeg7:timePointType" use="optional"/>
   <attribute name="creationPlace" type="string" use="optional"/>
   <attribute name="creationTool" type="string" use="optional"/>
   <attribute name="creator" type="string" use="optional"/>
   <attribute name="owner" type="string" use="optional"/>
   <attribute name="copyright" type="string" use="optional"/>
</complexType>
```

Within AXMEDIS it has to be identified, how cryptographic hash values can be integrated best into MPEG-7 and the relationship with MPEG-21.

Module Profile	
Descriptor Extractor Prototype for General Digital Resources	
Executable or Library(Support)	Library
Single Thread or Multithread	Multithreaded
Language of Development	C++
Responsible Name	Martin Schmucker (FHGIGD)
Responsible Partner	FHGIGD

Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
None	None	None
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
None	C++	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
BeeCrypt (?)		LGPL
Botan (?)		BSD-style
CryptLib (?)		GPL and standard commercial license

3 Appendix: IDL Interface

3.1 Interface of the Metadata Extractors for Text (IDL, WSDL)

3.1.1 Interfaces to descriptors extraction from text documents (IDL)

```
module DescriptorsExtractorForDocuments {

   /** Languages supported by the AXMEDIS framework
    */
   enum LANGUAGES { de, en, es, fr, it };

   /** ****** TO BE REVIEWED ***********
    * DocumentFile is the input format of the document.
    * It's defined as a octet sequence just for being the
    * most general (BLOB format). It will be changed as soon as
    * input formats and methods specifications for plugins will be defined.
    * It is assumed that this module deals only with plain text docs:
    * conversions (if needed) must be done before using classes
    * and methods of this module.
    */
   typedef sequence <octet> DocumentFile;

   /** Just a sequence of string containing a keyword per element.
    */
   typedef sequence <string> Keywords;

   /** Output type of text tokenization */
   typedef sequence <wstring> TextTokens;

   /** This interface provides the description of classes that can
    *  guess the main language of a document.
    *  An estimation of the guessing precision is given.
    */
```

```
interface LanguageGuesser {

   /** Sets the document to be analyzed */
   void setDocument(in DocumentFile doc2analyze);
   /** Retrieves the main language of the document */
   LANGUAGES getLanguage();
   /** An estimation of the language guessing */
   //readonly attribute double language_guessing_score;

};

/** This interface provides the description of classes that are able to
 *  extracts content descriptors from a document.
 */
interface DescriptorExtractor {

   /** Sets the document to be analyzed */
   void setDocument(in DocumentFile doc2analyze);
   /** Retrieves keywords using internal statistics, see algorithm A.
    * max_kw_number is the maximum number of keywords requested, default 3.
    */
   Keywords getKWFromInternalStatistics(in unsigned short max_kw_number);
   /** Retrieves keywords using statistics based on corpora comparisons,
    * see algorithm B.
    * max_kw_number is the maximum number of keywords requested, default 3.
    */
   Keywords getKWFromComparisons(in unsigned short max_kw_number);
   /** Retrieves keywords using statistics based on semantic analysis,
    * see algorithm C
    * max_kw_number is the maximum number of keywords requested, default 3.
    */
   Keywords getKWFromSemanticAnalysis(in unsigned short max_kw_number, in LANGUAGES
out_lang);
   /** Retrieves the domain of the document */
   wstring getDomain(in LANGUAGES out_lang);

   /** Delete all whitespaces and formatting chars
    *  from the plain text document, then returns it.
    */
   wstring getTextWithoutSpaces();
   /** Delete all punctuation chars from the plain
    *  text document, then returns it.
    */
   wstring getTextWithoutPunctuation();
   /** Delete all consonants chars from the plain text
    *  document, then returns it.
    */
   wstring getOnlyVowels();
   /** Delete all vowels chars from the plain text
    *  document, then returns it.
    */
   wstring getOnlyConsonants();
   /** Get the byte size of the plain text file */
   unsigned long getSizeInByte();
   /** Get the chars number of the plain text file */
   unsigned long getCharsNumber();
   /** Get the words of the plain text file */
   TextTokens getWords();
   /** Get the lines of the plain text file */
   TextTokens getLines();
   /** Get the periods of the plain text file */
   TextTokens getPeriods();
   /** Get the paragraphs of the plain text file */
   TextTokens getParagraphs();
   /** Get the graphic words number of the plain text file
    */
   unsigned long getWordsNumber();
```

```
      /** Get lines number of the plain text file */
      unsigned long getLinesNumber();
      /** Get periods number of the plain text file */
      unsigned long getPeriodsNumber();
      /** Get the paragraphs number of the plain text file
       */
      unsigned long getParagraphsNumber();
      /** Get the character encoding format of the plain text file.
       * ******** TO BE ASSESSED ************
       * Define which encodings are supported
       * ***********************************
       */
      string getCharacterEncoding();
   };
};
```

3.1.2 Interfaces to descriptors extraction from text documents (WSDL)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Marco Fabbri (LABLITA)
-->
<definitions                                    xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/"
xmlns:ns="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/types/"
targetNamespace="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/">
   <types>
      <xs:schema
targetNamespace="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/types/
"         xmlns="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/types/"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
         <xs:element name="DocumentFile" type="xs:hexBinary"/>
         <xs:element name="LANGUAGES">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="de"/>
                  <xs:enumeration value="en"/>
                  <xs:enumeration value="es"/>
                  <xs:enumeration value="fr"/>
                  <xs:enumeration value="it"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Max_KW_Number">
            <xs:simpleType>
               <xs:restriction base="xs:integer">
                  <xs:minInclusive value="0"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:schema>
      <xs:schema
targetNamespace="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/types/
"         xmlns="http://axmedis.org/wsdl/DescriptorExtractorforTextandDocuments/types/"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
         <xs:complexType name="Keywords">
            <xs:sequence>
               <xs:element name="keyword" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="TextTokens">
            <xs:sequence>
               <xs:element name="token" type="xs:string"/>
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:schema>
      </types>
      <message name="doc2analyze">
        <part name="parameters" element="ns:DocumentFile"/>
      </message>
      <message name="lang_out">
        <documentation>Languages supported by the AXMEDIS framework</documentation>
        <part name="parameters" element="ns:LANGUAGES"/>
      </message>
      <message name="kw">
        <documentation>Just a sequence of string containing a keyword per
element</documentation>
        <part name="parameters" type="ns:Keywords"/>
      </message>
      <message name="max_kw_number">
        <part name="parameters" element="ns:Max_KW_Number"/>
      </message>
      <message name="domain_out">
        <part name="parameters" type="xs:string"/>
      </message>
      <message name="text_out">
        <part name="parameters" type="xs:string"/>
      </message>
      <message name="number_out">
        <part name="parameters" type="xs:unsignedLong"/>
      </message>
      <message name="toks_out">
        <documentation>Output type of text tokenization</documentation>
        <part name="parameters" type="ns:TextTokens"/>
      </message>
      <message name="semantics_in">
        <part name="lang" element="ns:LANGUAGES"/>
        <part name="num" type="xs:integer"/>
      </message>
      <portType name="LanguageGuesser">
        <documentation>This interface provides the description of classes that can
          guess the main language of a document.
        </documentation>
        <operation name="setDocument">
          <documentation>Sets the document to be analyzed</documentation>
          <input message="tns:doc2analyze"/>
        </operation>
        <operation name="getLanguage">
          <documentation>Retrieves the main language of the document</documentation>
          <output message="tns:lang_out"/>
        </operation>
      </portType>
      <portType name="DescriptorExtractor">
        <documentation>This interface provides the description of classes that can
          guess the main language of a document.
        </documentation>
        <operation name="setDocument">
          <documentation>Sets the document to be analyzed</documentation>
          <input message="tns:doc2analyze"/>
        </operation>
        <operation name="getKWFromInternalStatistics">
          <documentation>Retrieves keywords using internal statistics, see algorithm A.
          max_kw_number is the maximum number of keywords requested, default
3.</documentation>
          <input message="tns:max_kw_number"/>
          <output message="tns:kw"/>
        </operation>
        <operation name="getKWFromComparisons">
          <documentation>Retrieves keywords using statistics based on corpora
comparisons, see algorithm B.
```

```
        max_kw_number is the maximum number of keywords requested, default
3.</documentation>
        <input message="tns:max_kw_number"/>
        <output message="tns:kw"/>
      </operation>
      <operation name="getKWFromSemanticAnalysis">
        <documentation>Retrieves keywords using statistics based on semantic analysis,
see algorithm C
        max_kw_number is the maximum number of keywords requested, default
3.</documentation>
        <input message="tns:semantics_in"/>
        <output message="tns:kw"/>
      </operation>
      <operation name="getDomain">
        <documentation>Retrieves the domain of the document</documentation>
        <input message="tns:lang_out"/>
        <output message="tns:domain_out"/>
      </operation>
      <operation name="getTextWithoutSpaces">
        <documentation>Delete all whitespaces and formatting chars
        from the plain text document, then returns it.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getTextWithoutPunctuation">
        <documentation>Delete all punctuation chars from the plain
        text document, then returns it.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getOnlyVowels">
        <documentation>Delete all consonants chars from the plain text
        document, then returns it.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getOnlyConsonants">
        <documentation>Delete all vowels chars from the plain text
        document, then returns it.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getSizeInByte">
        <documentation>Get the byte size of the plain text file.</documentation>
        <output message="tns:number_out"/>
      </operation>
      <operation name="getCharsNumber">
        <documentation>Get the chars number of the plain text file.</documentation>
        <output message="tns:number_out"/>
      </operation>
      <operation name="getWords">
        <documentation>Get the words of the plain text file.</documentation>
        <output message="tns:toks_out"/>
      </operation>
      <operation name="getLines">
        <documentation>Get the lines of the plain text file.</documentation>
        <output message="tns:toks_out"/>
      </operation>
      <operation name="getPeriods">
        <documentation>Get the periods of the plain text file.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getParagraphs">
        <documentation>Get the paragraphs of the plain text file.</documentation>
        <output message="tns:text_out"/>
      </operation>
      <operation name="getWordsNumber">
        <documentation>Get the graphic words number of the plain text
file.</documentation>
        <output message="tns:number_out"/>
      </operation>
```

```
      <operation name="getLinesNumber">
         <documentation>Get lines number of the plain text file.</documentation>
         <output message="tns:text_out"/>
      </operation>
      <operation name="getPeriodsNumber">
         <documentation>Get periods number of the plain text file.</documentation>
         <output message="tns:text_out"/>
      </operation>
      <operation name="getParagraphsNumber">
         <documentation>Get    the    paragraphs    number    of    the    plain    text
file.</documentation>
         <output message="tns:text_out"/>
      </operation>
      <operation name="getCharacterEncoding">
         <documentation>Get the character encoding format of the plain text file.
         * ******** TO BE ASSESSED ************
         * Define which encodings are supported
         * ***********************************
         </documentation>
         <output message="tns:text_out"/>
      </operation>
   </portType>
   <binding name="bind_LanguageGuesser" type="tns:LanguageGuesser">
      <operation name="setDocument">
         <input/>
      </operation>
      <operation name="getLanguage">
         <output/>
      </operation>
   </binding>
   <binding name="bind_DescriptorExtractor" type="tns:DescriptorExtractor">
      <operation name="setDocument">
         <input/>
      </operation>
      <operation name="getKWFromInternalStatistics">
         <input/>
         <output/>
      </operation>
      <operation name="getKWFromComparisons">
         <input/>
         <output/>
      </operation>
      <operation name="getKWFromSemanticAnalysis">
         <input/>
         <output/>
      </operation>
      <operation name="getDomain">
         <input/>
         <output/>
      </operation>
      <operation name="getTextWithoutSpaces">
         <output/>
      </operation>
      <operation name="getTextWithoutPunctuation">
         <output/>
      </operation>
      <operation name="getOnlyVowels">
         <output/>
      </operation>
      <operation name="getOnlyConsonants">
         <output/>
      </operation>
      <operation name="getSizeInByte">
         <output/>
      </operation>
      <operation name="getCharsNumber">
         <output/>
```

```
      </operation>
      <operation name="getWords">
         <output/>
      </operation>
      <operation name="getLines">
         <output/>
      </operation>
      <operation name="getPeriods">
         <output/>
      </operation>
      <operation name="getParagraphs">
         <output/>
      </operation>
      <operation name="getWordsNumber">
         <output/>
      </operation>
      <operation name="getLinesNumber">
         <output/>
      </operation>
      <operation name="getPeriodsNumber">
         <output/>
      </operation>
      <operation name="getParagraphsNumber">
         <output/>
      </operation>
      <operation name="getCharacterEncoding">
         <output/>
      </operation>
   </binding>
   <service name="serviceName"/>
</definitions>
```

3.2 Interface of the Fingerprint Extractors for Text (IDL, WSDL)

3.2.1 Interfaces to Fingerprint extraction from text documents (IDL)

```
module FPExtractorForDocuments {

   /** ****** TO BE REVIEWED ***********
    * DocumentFile is the input format of the document.
    * It's defined as a octet sequence just for being the
    * most general (BLOB format). It will be changed as soon as
    * specifications for input formats and methods (for plug-ins in general)
    * will be defined .
    */
   typedef sequence <octet> DocumentFile;

   /** Just a sequence of char to store hash signatures */
   typedef sequence <char> HashSignature;

   /** Enumeration of available hashing algorithms.
    *  The list can be changed according to AXMEDIS framework needs.
    */
   enum HASHING_ALGORITHMS { md5, sha1, sha2 };

   /** This interface provides the description of classes with which
    *  the hash signature of a document can be calculated.
    *  Different classes must be implemented in order to menage
    *  binary and plain text file.
    */
   interface FPExtractorForDocs {
      /** Sets the document to be analyzed */
      void setDocument(in DocumentFile doc2analyze);
      /** The hash algorithm used to calculate the hash signature */
      attribute HASHING_ALGORITHMS hashAlgorithm;
      /** Gets the hash signature of the document */
```

```
      HashSignature getHashSignature();
   };
};
```

3.2.2 Interfaces to fingerprint extraction from text documents (WSDL)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Marco Fabbri (LABLITA)
-->
<definitions                                    xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://axmedis.org/wsdl/FingerprintExtractorforTextandDocuments/"
xmlns:ns="http://axmedis.org/wsdl/FingerprintExtractorforTextandDocuments/types/"
targetNamespace="http://axmedis.org/wsdl/FingerprintExtractorforTextandDocuments/">
   <types>
      <xs:schema
targetNamespace="http://axmedis.org/wsdl/FingerprintExtractorforTextandDocuments/types
/"       xmlns="http://axmedis.org/wsdl/FingerprintExtractorforTextandDocuments/types/"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
         <xs:element name="DocumentFile" type="xs:hexBinary"/>
         <xs:element name="HASH_ALGORITHM">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="md5"/>
                  <xs:enumeration value="sha1"/>
                  <xs:enumeration value="sha2"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="HashSignature" type="xs:string"/>
      </xs:schema>
   </types>
   <message name="doc2analyze">
      <part name="parameters" element="ns:DocumentFile"/>
   </message>
   <message name="hash_out">
      <part name="parameters" element="ns:HASH_ALGORITHM"/>
   </message>
   <message name="hash_in">
      <part name="parameters" element="ns:HASH_ALGORITHM"/>
   </message>
   <message name="signature_out">
      <part name="parameters" element="ns:HashSignature"/>
   </message>
   <portType name="FPExtractorForDocs">
      <documentation>This interface provides the description of classes with which
    the hash signature of a document can be calculated.
    Different classes must be implemented in order to menage
    binary and plain text file.
   </documentation>
      <operation name="setDocument">
         <documentation>Sets the document to be analyzed</documentation>
         <input message="tns:doc2analyze"/>
      </operation>
      <operation name="getHashAlgorithm">
         <documentation>Gets the hash algorithm used</documentation>
         <output message="tns:hash_out"/>
      </operation>
      <operation name="setHashAlgorithm">
         <documentation>Sets the hash algorithm to be used</documentation>
         <input message="tns:hash_in"/>
      </operation>
```

```
      <operation name="getHashSignature">
         <documentation>Gets the hash signature of the document</documentation>
         <output message="tns:signature_out"/>
      </operation>
   </portType>
   <binding name="bind_ FPExtractorForDocs" type="tns:FPExtractorForDocs">
      <operation name="setDocument">
         <input/>
      </operation>
      <operation name="getHashSignature">
         <output/>
      </operation>
      <operation name="getHashAlgorithm">
         <output/>
      </operation>
      <operation name="setHashAlgorithm">
         <input/>
      </operation>
   </binding>
   <service name="serviceName"/>
</definitions>
```

4 Appendix: AXMEDIS Parameter and Content Descriptors Description (FHGIGD, DSI, EPFL, DIPITA)

4.1 Parameter Description (XML Schema) (FHGIGD, DSI, EPFL, DIPITA)

As parameters are described in XML, for their description a limited subset of XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, [XMLSchema]

4.2 Content Descriptors Description (XML Schema) (FHGIGD, DSI, EPFL, DIPITA)

MPEG-7 Descriptors compliant to MPEG-7 DDL ISO/IEC 15938. The content descriptors description has to be compliant with MPEG-7 (for further details see also next section). A reference software is available at:
http://www.lis.ei.tum.de/research/bv/topics/mmdb/mpeg7.html

5 Appendix: ISO/IEC 15938 Information Technology - Multimedia Content Description Interface (FHGIGD)

The MPEG-7 standard also known as "Multimedia Content Description Interface" aims at providing standardized core technologies allowing description of audiovisual data content in multimedia environments. In order to achieve this broad goal, MPEG-7 will standardize:

- Descriptors (D): representations of Features, that define the syntax and the semantics of each feature representation,
- Description Schemes (DS), that specify the structure and semantics of the relationships between their components, which may be both Ds and DSs,
- A Description Definition Language (DDL), to allow the creation of new DSs and, possibly, Ds and to allows the extension and modification of existing DSs,
- System tools, to support multiplexing of description, synchronization issues, transmission mechanisms, file format, etc.

The MPEG-7 standard consists of the following parts, under the general title Information Technology - Multimedia Content Description Interface:

- **Part 1: Systems**. Architecture of the standard, tools that are needed to prepare MPEG-7 Descriptions for efficient transport and storage, and to allow synchronization between content and descriptions. Also tools related to managing and protecting intellectual property.
- **Part 2: Description Definition Language (DDL)**. Language for defining new DSs and perhaps eventually also for new Ds, binary representation of DDL expressions.
- **Part 3: Visual**. Visual elements (Ds and DSs).
- **Part 4: Audio**. Audio elements (Ds and DSs).
- **Part 5. Multimedia Description Schemes**. Elements (Ds and DSs) that are generic, i.e. neither purely visual nor purely audio.
- **Part 6. Reference Software**. Software implementation of relevant parts of the MPEG-7 Standard.
- **Part 7. Conformance**. Guidelines and procedures for testing conformance of MPEG-7 implementations.

5.1 Part 2: Description Definition Language (ISO/IEC 15938-2)

The intention of the MPEG-7 DDL, as described in Part 2, is to enable MPEG-7 users and developers to:

- create valid MPEG-7 description schemes and descriptors;
- develop tools such as editors and parsers for processing descriptions, description schemes and descriptors;
- generate refinements, extensions and modifications to the DDL.

It describes the various features of the DDL. It defines the syntax of the DDL constructs and datatypes and provides (non-normative) examples which illustrate their application.

A (non-normative) prototype DDL parser tool is being developed by members of the DDL AHG, which validates the syntax of description scheme and descriptor definitions as well as instantiations of corresponding descriptions. The parser will need to parse both pure XML Schema schemes and descriptions as well as MPEG-7 extensions to XML Schema Language.

5.2 Part 3: Visual (ISO/IEC 15938-3)

Part 3 contains the visual elements (Descriptors and Description Schemes) that are considered for being part of the standard. All these Descriptive Structures are classified according to the types of visual features they describe. For each Descriptive Structure, there is one corresponding section in the document. The section specifies textual and binary syntax and semantics of the structures.

This part specifies tools for description of visual content, including still images, video and 3D models. These tools are defined by their syntax in DDL and binary representations and semantics associated with the syntactic elements. They enable description of the visual features of the visual material, such as color, texture, shape and motion, as well as localization of the described objects in the image or video sequence.

The basic structure description tools include five supporting tools of visual descriptions. They are categorized into two groups, descriptor containers and basic supporting tools. The former consists of three datatypes, GridLayout providing efficient representations of visual features on grids, TimeSeries representing temporal arrays of several descriptions, and MultipleView describing a 3D object using several pictures captured from different view angles. The latter contains two tools, Spatial2DCoordinateSystem used to specify the 2D coordinate system and TemporalInterpolation indicating the interpolation method between two samples on a time axis.

The remaining description tools, except for the FaceRecognition descriptor, are associated with visual features and are grouped into five feature categories: Color, Texture, Shape, Motion and Localization.

The color description tools include four color descriptors to represent different aspects of color features: representative colors (DominantColor), color distribution (ScalableColor), spatial distribution of colors (ColorLayout and ColorStructure). It also contains two supporting tools, ColorSpace and ColorQuantization used in DominantColor and an extension of ScalableColor to a group of frames or pictures (GoFGoPColor). All the color descriptors can be extracted from arbitrarily shaped regions.

The texture description tools facilitate browsing (TextureBrowsing) and similarity retrieval (HomogeneousTexture and EdgeHistogram) using the texture of a still or moving image region. All the texture descriptors can be extracted from arbitrarily shaped regions.

The shape description tools include two descriptors that characterize different shape features of a 2D object or region. The RegionShape descriptor captures the distribution of all pixels within a region and the Contour Shape descriptor characterizes the shape properties of the contour of an object. The Shape3D descriptor provides an intrinsic shape characterization of 3D mesh models.

The motion description tools include four descriptors that characterize various aspects of motion. The CameraMotion descriptor specifies a set of basic camera operations such as, for example, panning and tilting. The motion of a key point (pixel) from a moving object or region can be characterized by the MotionTrajectory descriptor. The ParametricMotion descriptor characterizes an evolution of an arbitrarily shaped region over time in terms of a 2D geometric transformation. Finally, the MotionActivity descriptor captures the pace of the motion in the sequence, as perceived by the viewer. All motion descriptors except for CameraMotion can be extracted from arbitrarily shaped regions.

The localization description tools can be used to indicate regions of interest in the spatial (RegionLocator) and spatio-temporal (SpatioTemporalLocator) domains.

The FaceRecognition descriptor is not associated with any particular visual feature and can be used to describe a human face for applications requiring the matching and retrieval of face images.

5.3 Part 4: Audio (ISO/IEC 15938-4)

Audio is applicable to all forms of audio content. The encoding format or medium of the said audio is not limited in any way, and may include audio held in an analogue medium such as magnetic tape or optical film. The content of the audio is not limited within or without music, speech, sound effects, soundtracks, or any mixtures thereof.

The tools listed in Part 4 of the International Standard are applicable to both audio in isolation and to audio associated with video.

The specific tools provided within the Audio portion of the standard are designed to work in conjunction with the Multimedia Description Schemes that apply to both audio and video. Because of the “toolbox” nature of the standard, the most appropriate tools from the different parts of the standard may be mixed, within the constraints of the DDL.

The MPEG-7 Audio tools are applicable to two general areas: low-level audio description and application-driven description.

The Audio Framework tools are applicable to general audio, without regard to the specific content carried by the encoded signal. The Scalable Series provides general capabilities for multi-level sampled data. The Audio Description Framework defines specific descriptors for use with the Scalable Series or with Audio Segments, which has properties inherited from the general Segment described in the Multimedia Description Schemes part of the standard. The Silence Descriptor works with the Segment descriptor, and is applicable across all possible audio signals.

The Application-driven description tools are applicable to specific types of content within audio. The specific domains are well documented within the introduction to each sub-clause. The audio domains encompassed by the various MPEG-7 Audio tools are speech, sound effects, musical instruments, melodies within music and general audio recognition. These specialised tools may be employed in conjunction with the other tools within the standard.

5.4 Part 5: Multimedia Description Scheme (ISO/IEC 15938-5)

Part 5 describes the MDS description tools. In the sequel, each description tool is described in two or three normative sections:

- Syntax: Normative DDL specification of the Ds or DSs.
- Binary Syntax: Normative binary representation in case a specific binary representation has been designed.
- Semantic: Normative definition of the semantics of all the components of the corresponding D or DS.
- And optionally, by an informative section dealing with examples.

6 Bibliography (FHGIGD)

[MPEG-7] MPEG, MPEG-7 Overview, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[RCF2045] RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, http://www.faqs.org/rfcs/rfc2045.html

[RCF2046] RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, http://www.faqs.org/rfcs/rfc2046.html

[RCF2077] RFC 2077 - The Model Primary Content Type for Multipurpose Internet Mail Extensions, http://www.faqs.org/rfcs/rfc2077.html

[WIHF] Wikipedia, “hash function”, http://en.wikipedia.org/wiki/Hash_function

[XMLSchema] XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/#schema