INTERNATIONAL STANDARD

ISO/IEC 15938-6

First edition
2003-07-01

Information technology — Multimedia content description interface —

Part 6: Reference software

Technologies de l'information — Interface de description du contenu multimédia —

Partie 6: Logiciel de référence

Reference number
ISO/IEC 15938-6:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 15938-6 was prepared by Technical Committee ISO/IEC/TC JTC1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 15938 consists of the following parts, under the general title *Information technology — Multimedia content description interface*:

— *Part 1: Systems*

— *Part 2: Description definition language*

— *Part 3: Visual*

— *Part 4: Audio*

— *Part 5: Multimedia description schemes*

— *Part 6: Reference software*

— *Part 7: Conformance testing*

— *Part 8: Extraction and use of MPEG-7 descriptions*

Introduction

This International Standard, also known as “Multimedia Content Description Interface”, provides a standardized set of technologies for describing multimedia content. International Standard addresses a broad spectrum of multimedia applications and requirements by providing a metadata system for describing the features of multimedia content.

The following are specified in this International Standard:

- **Description schemes (DS)** describe entities or relationships pertaining to multimedia content. Description Schemes specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.
- **Descriptors (D)** describe features, attributes, or groups of attributes of multimedia content.
- **Datatypes** are the basic reusable datatypes employed by Description Schemes and Descriptors.
- **Systems tools** support delivery of descriptions, multiplexing of descriptions with multimedia content, synchronization, file format, and so forth.

This International Standard is subdivided into eight parts:

Part 1 – Systems: specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and descriptions.

Part 2 – Description definition language: specifies the language for defining the standard set of description tools (DSs, Ds, and datatypes) and for defining new description tools.

Part 3 – Visual: specifies the description tools pertaining to visual content.

Part 4 – Audio: specifies the description tools pertaining to audio content.

Part 5 – Multimedia description schemes: specifies the generic description tools pertaining to multimedia including audio and visual content.

Part 6 – Reference software: provides a software implementation of the standard.

Part 7 – Conformance testing: specifies the guidelines and procedures for testing conformance of implementations of the standard.

Part 8 – Extraction and use of MPEG-7 descriptions: provides guidelines and examples of the extraction and use of descriptions.

This part of ISO/IEC 15938 contains simulation software for tools defined in parts 1, 2, 3, 4 and 5 of ISO/IEC 15938. This software has been derived from the verification models used in the process of developing the International Standard.

Where multimedia content extraction or multimedia content description software is provided, attention is called to the fact that these software modules are provided for the purpose of creating bit streams of descriptors and description schemes with normative syntax. The performance of these software tools should not be taken as indicative of that which can be obtained from implementations where quality and computational optimization are given priority. The techniques used for extracting descriptors or deriving description schemes are not specified by this document. This information can be found in the corresponding sections of part 1-5.

Information technology — Multimedia content description interface —

Part 6: Reference software

1 Scope

This International Standard operates on and generates conformant bit streams. This International Standard provides a specific implementation that behaves in a conformant manner. In general, other implementations that conform to ISO/IEC 15938 are possible that do not necessarily use the algorithms or the programming techniques of the reference software.

The software contained in this part of ISO/IEC 15938 is known as experimentation software (XM) and is divided into five categories:

a) **Binary format for MPEG-7 (BiM)**. This software converts DDL (XML) based descriptions to the Binary format of MPEG-7 and vice versa as explained in Clause 5 of this document.

b) **DDL parser and DDL validation parser**. The components of this software module are specified in Clause 6 of this document.

c) **Visual descriptors**. This software creates standard visual descriptions from associated (visual) media content as explained in Clause 7 of this document. The techniques used for extracting descriptors are informative, and the quality and complexity of these extraction tools has not been optimized.

d) **Audio descriptors**. This software creates standard descriptions from associated (audio) media content as explained in Clause 8 of this document. The techniques used for extracting descriptors are informative, and the quality and complexity of these extraction tools has not been optimized.

e) **Multimedia description schemes**. This software modules provide standard descriptions of Multimedia Description Schemes as specified in Clause 9 of this document.

2 Symbols and abbreviated terms

For the purposes of this part of ISO/IEC 15938, the following symbols and abbreviated terms apply:

AV: Audio-visual

CS: Coding Scheme

D: Descriptor

Ds: Descriptors

DCT: Discrete Cosine Transform

DDL: Description Definition Language

DS: Description Scheme

DSs: Description Schemes

ISO: International Organization for Standardization

MDS: Multimedia Description Schemes

MPEG: Moving Picture Experts Group

MPEG-7: Multimedia Content Description Interface Standard (see ISO/IEC 15938)

XML: Extensible Markup Language

3 Copyright disclaimer for software modules

Each source code module in this specification contains copyright disclaimer which shall not be removed from the source code module.

In the text of each copyright disclaimer, <MPEG standard> is replaced with a reference to its associated specification, e.g. MPEG-7 System (ISO/IEC 15938-1), MPEG-7 Video (ISO/IEC 15938-3), MPEG-7 Audio (ISO/IEC 15938-4), MPEG-7 Multimedia Description Scheme (ISO/IEC 15938-5).

"This software module was originally developed by <FN1> <LN1> (<CN1>) and edited by <FN2> <LN2> (<CN2>), <FN3> <LN3> (<CN3>), … in the course of development of the <MPEG standard>. This software module is an implementation of a part of one or more <MPEG standard> tools as specified by the <MPEG standard>. ISO/IEC gives users of the <MPEG standard> free license to this software module or modifications thereof for use in hardware or software products claiming conformance to the <MPEG standard>. Those intending to use this software module in hardware or software products are advised that its use may infringe existing patents. The original developer of this software module and his/her company, the subsequent editors and their companies, and ISO/IEC have no liability for use of this software module or modifications thereof in an implementation. Copyright is not released for non <MPEG standard> conforming products. CN1 retains full right to use the code for his/her own purpose, assign or donate the code to a third party and to inhibit third parties from using the code for non <MPEG standard> conforming products. This copyright notice must be included in all copies or derivative works. Copyright ©200_".

<FN>=First Name, <LN>=Last Name, <CN>=Company Name

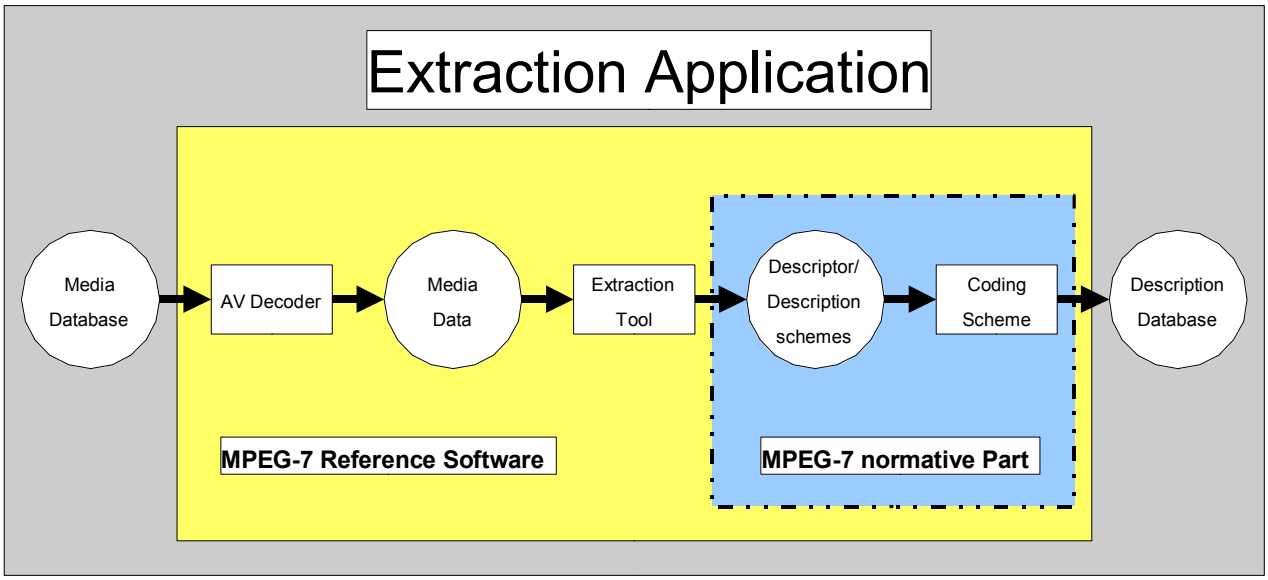
4 XM software architecture

4.1 Block diagrams

In this section you will find some information about the XM software architecture. The block diagrams give short overviews, and introduce individual components of the XM software. The section also provides a list of the directory location for each module.

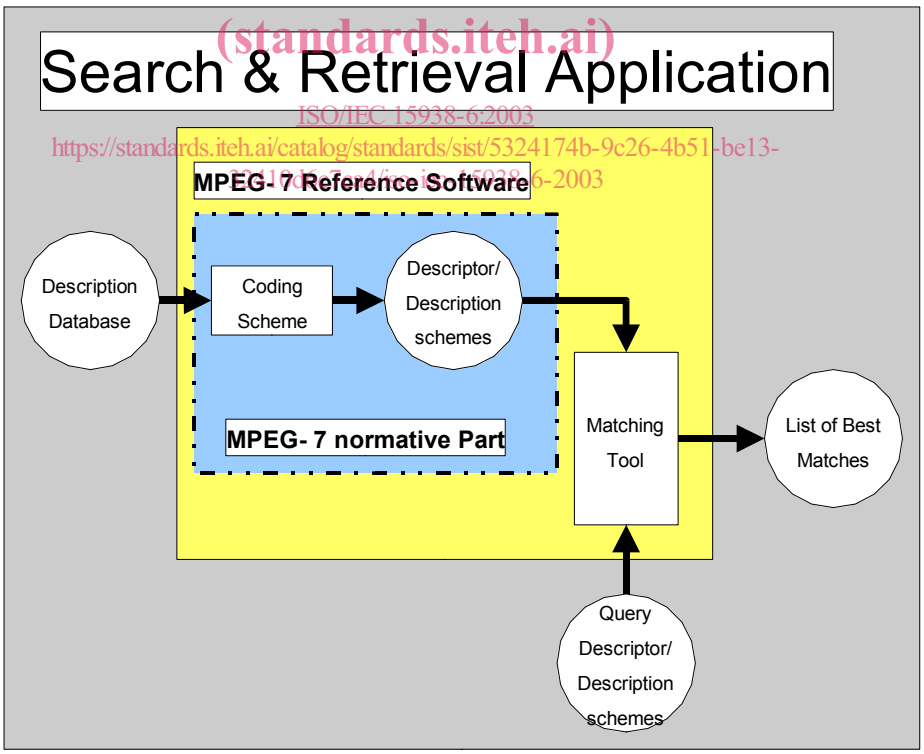
The composing elements of the MPEG-7 Reference Software are characterized by their functionality and by their interfaces. They can be configured according to what here is referred as "Key Applications". We can distinguish from the functional point of view:

- "Extraction Applications" (a description data base is built from a media data base)
- "Search and Retrieval Applications" (a description is compared with the descriptions in a database to find the one with the lowest distance)
- "Transcoding Applications" (a media data base is converted into another media data base basing on its description)



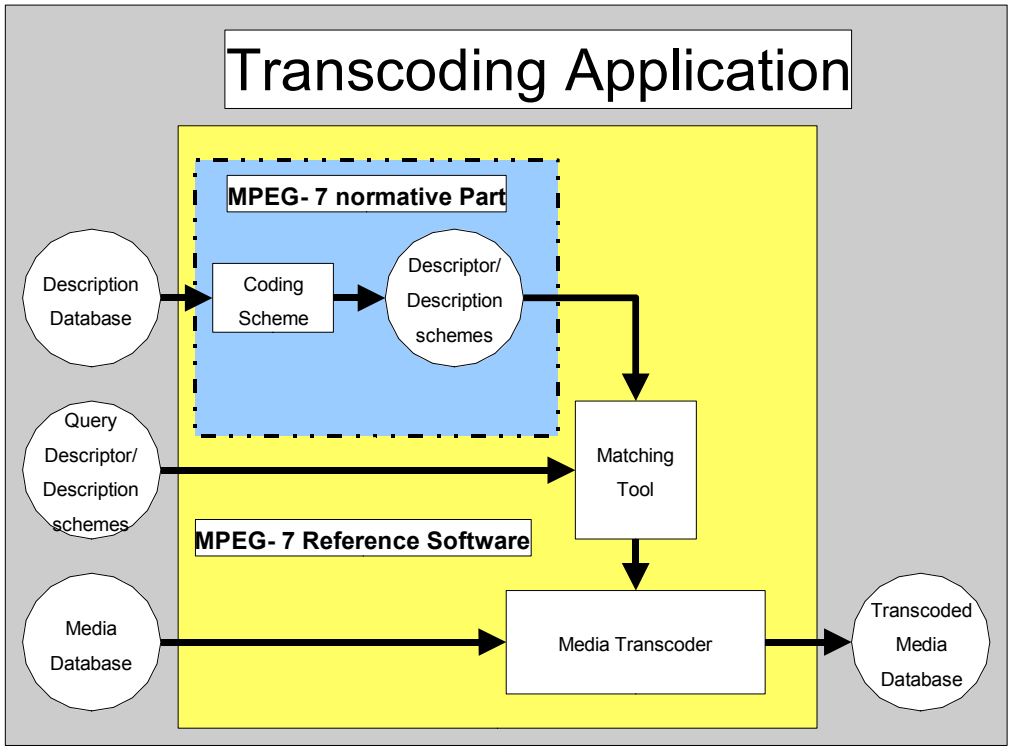
NOTE In the block diagram, boxes represent procedural parts, circles represent data structures.

Figure 1 — Schematic diagram of an "Extraction Application" using the XM reference software modules



NOTE In the block diagram, boxes represent procedural parts, circles represent data structures.

Figure 2 — Schematic diagram of a "Search and Retrieval Application" using the XM reference software modules



NOTE In the block diagram, boxes represent procedural parts, circles represent data structures.

Figure 3 — Schematic diagram of a "Transcoding Application" using the XM reference software modules

In the following, the blocks of the "Key Applications" are distinguished. For elements that are related to a specific descriptors or description schemes, the interface is given with the example of the DummyType (the XM integration template and not a normative descriptor).

4.1.1 Media database

The media database contains media files, which are supported as input file by the AV decoders. The database file which is read from a file, contains one media filename per line. From this media filename all additional input and output filenames can be derived.

4.1.2 AV decoders

The XM supports the following AV decoders:

- Still image decoders: ImageMagick (Ver.4.*-5.* linked as external library, not included in the XM reference software distribution)
- MPEG-1, MPEG-2 video decoders: (XM directory: Decoders/MPEG2Dec)
- MPEG-1 video motion vector extractor: (XM directory: Decoders/MPEG2Dec) (It can extract images and motion vectors)
- 3D Objects: (XM directory: Media) (It reads a 3D object for 3D shape descriptors)
- Key Points: (XM directory: Media) (It reads in a list of key points from a file).

4.1.3 Media data

This is the internal XM representation of the raw media data (one class with different structures depending on the media content type). The class description for media data can be found in the Media XM directory.

4.1.4 Extraction tools

Extraction tools are specific extraction methods defined for each Descriptor and Description Scheme. All these source file are available in the ExtractionUtilities XM directory. Extraction tools are not normative in the implementation but they must provide a valid description. The extraction tools extract the descriptions from media data. Because media data can be very big, the extraction is performed on time entities of the media, i.e., if the media is a video, the extraction is done frame by frame. The interface of the DummyType extraction tool (implementation template) is given below:

```
//=============================================================================
class DummyTypeExtractionTool: public DescriptorExtractor
{
  friend DummyTypeExtractionInterface;
public:
  // Null constructor
  DummyTypeExtractionTool();

  // Also connects the Descriptor (result memnory) to the extraction
  // If set to "0" it automatically creates the descriptor
  DummyTypeExtractionTool(DummyTypeDescriptorInterfaceABC
                *DummyType);

  // ID of object type
  virtual const UUID& GetObjectID(void);
  // Object type name
  virtual const char *GetName(void);

// This informs the extractor where the source data comes from
  virtual int SetSourceMedia(MultiMediaInterfaceABC* media);

// Pointer where the description is stored
  virtual DummyTypeDescriptorInterfaceABC*
    GetDescriptorInterface(void);
  virtual int SetDescriptorInterface(DummyTypeDescriptorInterfaceABC
                        *aDummyTypeDescriptorInterface);

  // initililaize descriptor and extraction process (input media must be known)
  virtual unsigned long InitExtracting(void);

  // performs extraction form input media frame by input media frame
  virtual unsigned long StartExtracting(void);

  // collects descriptor data after all input media frames were processed
  virtual unsigned long PostExtracting(void);

  // Extraction object must no be used, only its interface is allowd to
  // to be used. This function is to get the interface
  virtual DummyTypeExtractionInterfaceABC *GetInterface(void);

// access is allowed only by class factories for this
  // object.  This avoids having to duplicate the
  // ID definition in multiple locations.  In the future, we may
  // have to do this.  PLEASE DO NOT USE THESE UNLESS YOU ARE
  // IMPLEMENTING A CLASS FACTORY GENERATING THIS OBJECT
```

```
    static const UUID myID;
    static const char * myName;
  private:
    // Destructor is private to allow creation of
    // object only by using "new"
    virtual ~DummyTypeExtractionTool();

    DummyTypeExtractionInterface m_Interface;
    DummyTypeDescriptorInterfaceABC *m_DescriptorInterface;
    MultiMediaInterfaceABC* m_Media;

    // only used in this dummy type to show extraction function
    unsigned long m_FrameCnt;

  #ifdef __HasSubTypes /*include this section if sub descriptors exist,
                remove this section if no sub-descriptors exist*/
    SubDummyTypeAExtractionInterfaceABC *m_SubDummyTypeAExtraction;
    SubDummyTypeBExtractionInterfaceABC *m_SubDummyTypeBExtraction;
  #endif /* __HasSubTypes*/
    int m_DummyExtractionParameter;
  }; // End class
  //=============================================================================
```


4.1.5 Descriptors (Ds) and Description Schemes (DSs)

These modules implement the data structure of normative Descriptors and Description Schemes. Low level Video Descriptors are using a dedicated C++ class. This classes provide methods to access the elements of the normative descriptions. The source files are located in the Descriptors directory. All other normative Ds and DSs are using the GenericDS class located in the DescriptionSchemes directory. The GenericDS class does not implement the data structure in a dedicated way, but it is an interface to the XML parser library which controls the memory for the tree structure of the instantiated D or DS. The interface of the descriptors class is given for the DummyType descriptor (implementation template) below:

```
  /=============================================================================
  class DummyTypeDescriptor: public Descriptor
  {
  friend DummyTypeDescriptorInterface;
  public:
    DummyTypeDescriptor();

  #ifdef __HasSubTypes /*include this section if sub descriptors exist,
                remove this section if no sub-descriptors exist*/
    // constructor which also constructs and/or connects the descriptor object
    DummyTypeDescriptor(SubDummyTypeADecriptorInterfaceABC *aSubDummyTypeA,
               SubDummyTypeBDescriptorInterfaceABC *aSubDummyTypeB);
  #endif /* __HasSubTypes*/

  virtual const UUID& GetValueID(void);
    virtual const char* GetValueName(void);

    virtual const UUID& GetObjectID(void);
    virtual const char *GetName(void);
```