TECHNICAL REPORT

ISO/IEC TR 13818-5

First edition
1997-12-15

AMENDMENT 2
2005-04-01

Information technology — Generic coding of moving pictures and associated audio information —

Part 5: Software simulation

AMENDMENT 2: MPEG-2 IPMP reference software

Technologies de l'information — Codage générique des images animées et des informations sonores associées —

Partie 5: Simulation de logiciel

AMENDEMENT 2: Logiciel de référence MPEG-2 IPMP

Reference number
ISO/IEC TR 13818-5:1997/Amd.2:2005(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

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.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to ISO/IEC TR 13818-5:1997 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Generic coding of moving pictures and associated audio information —

Part 5:
Software simulation

AMENDMENT 2: MPEG-2 IPMP reference software

Replace 1.1 Scope with:

This part of ISO/IEC 13818 provides a C language software simulation of an encoder and decoder for Part 1 (systems), Part 2 (Video), Part 3 (Audio) and Part 11 (IPMP) of ISO/IEC 13818.

Insert the following additional references to 1.2:

ISO/IEC 13818-1:2000, *Information technology — Generic coding of moving pictures and associated audio information: Systems*

ISO/IEC 13818-11:2003, *Information technology — Generic coding of moving pictures and associated audio information — Part 11: Intellectual Property Management and Protection (IPMP) on MPEG-2 systems.*

Insert the following text as a new IPMP Modules section at the end of ISO/IEC 13818-5:

6 MPEG-2 IPMP Reference Software

6.1 Architecture

The codes of MPEG-2 IPMP reference software are developed with C++ on Windows Platform. However, the normative parts of the reference software don't use Windows specific libraries or functions. So it is quite easy to transport the codes to other platforms as well.

The major IPMP functionalities are encapsulated in a demultiplex filter, which uses Microsoft's DirectX technology. Doing that not only gives an easy terminal implementation, which needs to deal with demultplexing, decoding and rendering, but also brings the benefits of a clear terminal architecture. The general architecture of the software is shown in Figure 1. Based on this given architecture, a general data flow in a MPEG-2 IPMP terminal is described as below,

An input MPEG-2 Transport Stream (TS) is fed into a demultiplex filter, which decomposes the system stream into individual elementary streams. Before the TS stream is demultiplexed, the IPMP information carried inside are extracted and processed first. IPMP related actions are taken according to the processing results. For example, if the rights condition of the TS stream is not fulfilled, further processing will be terminated and user will be prompted of the rights failure. Furthermore, global initializations are also carried upon here, e.g., checking the availability the IPMP tools specified in the IPMP Tool List, retrieving the tools in case they don't exist in terminal, etc.

The TS stream is then demultplexed into elementary streams, typically video streams and audio streams. Before elementary streams are input to their respective decoder filters, IPMP processing is involved again, if one stream has IPMP control point set before the decoder, i.e., the IPMP descriptors associated with this stream explicitly request that data in the pre-decoder buffer need to be processed first by certain IPMP Tools, which are also specified in the IPMP descriptors. For example, decryption tools are needed to transcode encrypted data back to clear data before they can be correctly processed by decoder.

After the stream data go through the decoder filter, they are input to render filter. Before the render presents the data, IPMP is involved again to check whether the data in the pre-render buffer need to be processed by some IPMP tools first. For example, watermarking tools are used to extract signatures hidden in the stream data.

At last, the render filters present the clear stream data on user's terminals.

The internal communication between tools and terminal is in the form of IPMP messages, which are normative parts of the IPMP standard. All MPEG-2 IPMP compatible terminal and tools should understand the content of an IPMP message.

Message Routers act as an agent in the IPMP terminal. It takes charge of message sending and receiving between terminal and tools. If one tool needs to send message to another tool, the message also needs to go through terminal's Message Router and redirected to its destination. However, the interfaces between Message Router and IPMP Tools are not standardized. So the implementation can vary in different platforms and applications. This software just gives a reference implementation of the Message Router and reference model how the Message Router can facilitate the message delivering between IPMP Terminal and IPMP tools. Users are not limited by the reference implementation and free to extend from it by their wills.

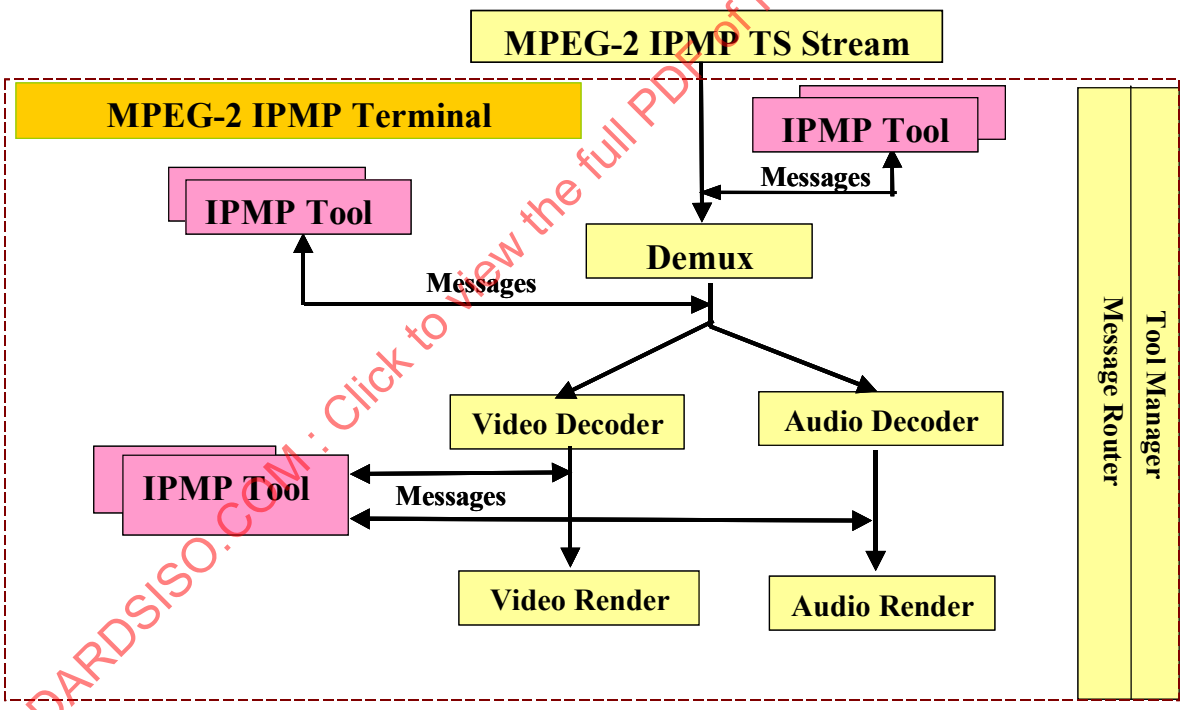

Figure AMD2-1 — MPEG-2 IPMP Terminal Architecture

Tool Manager acts as an agent to manage IPMP Tools in IPMP Terminal, e.g., loading IPMP Tool, connecting to IPMP Tool, unloading IPMP Tool, etc. Similar as Message Router, its interfaces are not standardized also. This software gives a reference implementation of the Tool Manager. Users of the software are not limited to extend its functionalities by their wills.

The following section introduces the implemented core components in details.

6.2 Core Components

6.2.1 IPMP Tool List – Normative

IPMP Tool List is carried in TS stream's PSI table. It gives the information of IPMP Tools to be used to process the given stream.

In the reference software, the modules related with IPMP Tool List are included in the demultiplex filter.

6.2.2 PMP Tool Container – Normative

IPMP Tool Container is carried in TS streams' PSI table. It is able to carry binary IPMP Tools. One possible scenario to use this feature is tool vendor wants to update or upgrade a updated tool with a later version.

In the reference software, the modules related with IPMP Tool Container are included in the demultiplex filter.

6.2.3 IPMP Rights Container - Normative

IPMP Rights Container is carried in TS streams' PSI table. It carries rights information associated with the host stream. The rights information should be extracted by the Terminal and processed by specified IPMP Tools.

In the reference software, the modules related with IPMP Rights Container are included in the demultiplex filter.

6.2.4 IPMP Descriptor – Normative

IPMP Descriptor is carried in TS stream's PMT table. Each descriptor is associated with an elementary stream, i.e., audio or video. Each descriptor specifies an IPMP Tool to be used to process the stream. The control point is given to tell terminal at what stage the tool to be applied.

In the reference software, the modules related with IPMP Descriptor are included in the demultiplex filter.

6.2.5 IPMP Messages – Normative

IPMP Messages are used to pass data and information between IPMP Terminal and IPMP Tools to facilitate IPMP processing.

There are 24 messages, listed in Table 1, defined in the MPEG-2 IPMP standard. All of them have been implemented in the reference software. They are encapsulated in an individual message library for user convenience.

Table AMD2-1 — MPEG-2 IPMP Messages

8-bit Tag Value	Symbolic Name
0x00	Forbidden
0x01	IPMP_OpaqueData_tag
0x02	IPMP_AudioWatermarkingInit_tag
0x03	IPMP_VideoWatermarkingInit _tag
0x04	IPMP_SelectiveDecryptionInit_tag
0x05	IPMP_KeyData _tag
0x06	IPMP_SendAudioWatermark_tag
0x07	IPMP_SendVideoWatermark _tag
0x08	IPMP_RightsData _tag
0x09	IPMP_Secure_Container_tag
0x0A	IPMP_AddToolNotificationListener_tag
0x0B	IPMP_RemoveToolNotificationListener_tag
0x0C	IPMP_InitAuthentication_tag
0x0D	IPMP_MutualAuthentication_tag
0x0E	IPMP_UserQuery_tag
0x0F	IPMP_UserQueryResponse_tag
0x10	IPMP_ToolParamCapabilitiesQuery_tag
0x11	IPMP_ToolParamCapabilitiesResponse_tag
0x12	IPMP_GetTools_tag
0x13	IPMP_GetToolsResponse_tag
0x14	IPMP_ConnectTool_tag
0x15	IPMP_DisconnectTool_tag
0x16	IPMP_NotifyToolEvent_tag
0x17	IPMP_CanProcess_tag
0x18	IPMP_TrustSecurityMetadata_tag
0x19– 0x3F	Reserved for Inter-device messages
0x40 – 0xCF	ISO Reserved
0xD0 – 0xFE	User Defined
0xFF	Forbidden

6.2.6 IPMP Stream - Normative

IPMP_Stream carries multiple IPMP Messages, which are packed by IPMP_StreamDataUpdate message. Multiple continuous IPMP_StreamDataUpdate messages can be concatenated together to form IPMP stream. One usage of this stream is to carry time-variant key data, and multiplex together with video or audio elementary streams. Upon receiving IPMP_StreamDataUpdate message, the Message Router will unpack the IPMP_StreamDataUpdate message and extract the inside messages for further processing.

The following three messages are used to facilitate the message delivery between terminal and tools. Table AMD2-2 lists what MPEG-2 IPMP has defined.

Table AMD2-2 — MPEG-2 IPMP Facilitating Messages

8-bit Tag Value	Symbolic Name
0x00	Forbidden
0x01	IPMP_MessageFromBitstream_tag
0x02	IPMP_DescriptorFromBitstream_tag
0x03	IPMP_MessageFromTool_tag
0x04 – 0xCF	ISO Reserved
0xD0 – 0xFE	User Defined
0xFF	Forbidden

In the reference software, all above messages have been implemented in the Message library.

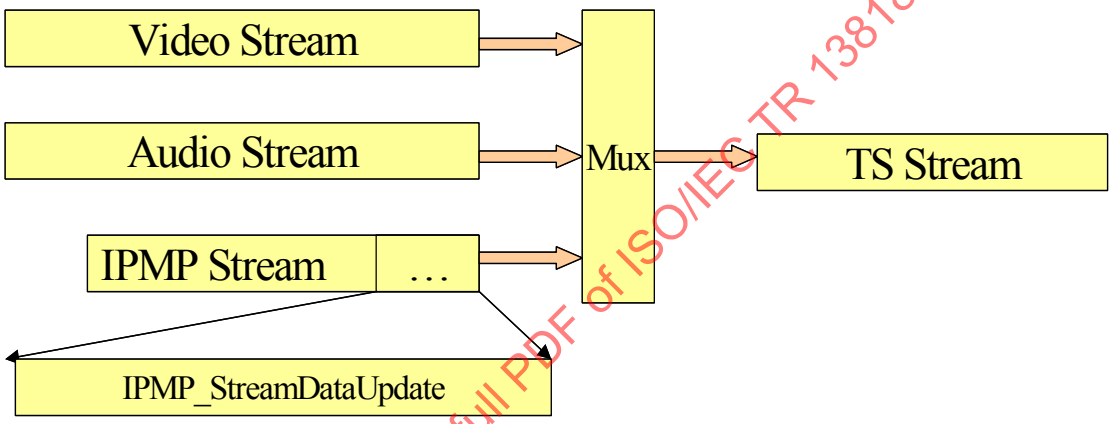

Figure AMD2-2 — IPMP Stream and other elementary streams

6.2.7 Message Router – Informative

Message Router manages the message delivery between Terminal and IPMP Tools. Because the practical applications could base on different platforms and have different context, the interfaces of Message Router exposed to IPMP Terminal and IPMP Tool aren't standardized. In this reference software, Class *CPSLMR* implements the major functionalities that a Message Router should have, that are receiving and delivering messages. The interfaces used by *CPSLMR* act just as reference for other users.

In the reference software, Message Router functions are encapsulated in class *CPSLMR,* which are used by the IPMPTerminal.

6.2.8 Tool Manager – Informative

Tool Manager manages the IPMP Tools used to process the input stream, e.g., loading the tools given in the IPMP Tool List, connecting the terminal with IPMP Tool, etc. Because the practical applications could base on different platforms and have different context, the interfaces of Tool Manager to IPMP Terminal and IPMP Tool aren't standardized. In this reference software, Class *IPMPToolManagerImp* implements the major functionalities that a Tool Manager should have. The interfaces used by *IPMPToolManagerImp* act just as reference for other users.

In the reference software, Tool Manager functions are encapsulated in class *IPMPToolManagerImp,* which are used by the IPMP Terminal.

6.2.9 IPMP Terminal – Informative

IPMP Terminal controls the whole process of input MPEG-2 IPMP stream. In the reference software implementation, the IPMP Terminal controls IPMP processing, demultiplexing, decoding and rendering.

In the reference software, IPMP Terminal's code is in *PlayWnd* project.

6.2.10 Multiplexer – Informative

For the convenience of MPEG-2 IPMP TS stream generation, the reference software also provides a multiplexer, a standalone application. It can multiplex MPEG-2 format video stream and MPEG-1 audio stream into MPEG-2 transport stream. According to the configuration, it can multiplex IPMP information into the TS, e.g., IPMP stream, IPMP Tool List, IPMP Tool Container, IPMP Rights Container, IPMP Descriptor, etc.

Furthermore, reference software provides a DES encryption/decryption tool to work together with the multiplexer. Users can encrypt elementary streams with this DES tool and multiplex the transformed stream into TS stream.

Please read the Readme.txt file under the *tsmuxer* project directory to get more details about how to use this application. Sample parameter file together with some descriptor data are provided to give users a startup example.

In the reference software, multiplexer's code is organized in *Tsmux* project.

6.2.11 IPMP Tool - Informative

One important principle of developing MPEG-2 IPMP standard is to achieve interoperability. The standard aims to provide a flexible framework to allow any compatible tools to interact each other, instead of the limitation of certain specific tools. Due to this consideration, there are no specific IPMP Tools defined. Any tools that process or governs the media stream data can be used as IPMP Tools as long as they can handle IPMP messages.

In the reference software, we provide 3 IPMP tools to demonstrate how IPMP Terminal and IPMP Tool interact each other. In the reference implementation, all IPMP tools are in the form of DLL libraries. They expose one and only one interface to the IPMP Terminal:

BOOL ToolInterface(unsigned int sender, unsigned int recipient, long size, byte message) (1)*

sender: the descriptor ID of message sender

recipient: the descriptor ID of message receiver

size: the size of this message

message: binary data of the message

Vice versa, the IPMP Terminal exposes the same interface the IPMP Tool as well to enable messages circulated between Terminal and Tools. The Terminal's public interface is,

BOOL MR_Interface(unsigned int Sender, unsigned int Recipient, long size, byte message) (2)*

It takes the same parameter as IPMP Tools.

In the reference software, the connection between Terminal and an IPMP Tool A is setup like following:

Step1: Terminal loads Tool A's Dll file

Step2: Terminal acquires Tool A's public interface, which is (1) as above

Step3: Terminal packs the pointer of its public interface, which is (2), in a IPMP_OpaqueData message, and send the message through (1) it gets from Step 2. In the reference code, we have a member function defined in the Message Router to do this:

Send_MRReceiveMessagePtr(unsigned int Recipient)

Recipient: the descriptor ID associated with the tool to receive this message

Step 4: Tool A receives a IPMP_OpaqueData message, which contains certain identifiers telling this is a interface pointer from Terminal. Tool A saves the pointer for future use.

By then, the connection setup between Terminal and Tool A is done. They both have known their counterpart's interface and are able to send/receive messages each other.

As have been said, this is only reference implementation. Other applications can do this in more constructive ways.

Three IPMP Tools provided by the reference softwares are,

- DES Tool

  This DES tool is able to encrypt/decrypt data with 64, 128 or 196 bits keys.

  In the reference software, the codes of this DES decryption tool are in project *DESDLL*.

- REL (Rights Expression Language) Tool

  This tool provides REL functions to validate Rights data in the form of MPEG-21 REL.

  In the reference software, the codes of this REL Tool are in project *RELTool*.

- Dummy Tool

  A dummy tool which simply accepts media streams from message router and returns without any processing. It can be easily extended to perform watermarking or encryption processing.

  In the reference software, the codes of this Dummy Tool are in project *DummyTool*.

6.3 Usage of the Reference Software

6.3.1 Building the Reference Software

The reference software codes are developed with Microsoft Visual C++ 6.0 on Windows platform. Please use Microsoft's Visual Studio to open each workspace introduced below and build them individually.

Workspace MPEG2SplitterFilter

The workspace *MPEG2SplitterFilter.dsw* contains 4 projects. They are,

- *PSL MPEG2 MessageInterface.dsp* under subdirectory MessageInterface.

This project contains the codes of IPMP Messages and the handling of the messages. The output of this project is *PSLMPEG2MessageInterface.lib*.

- *MPEG2Splitter.dsp* under subdirectory PSLMPEG2Lib.

This project contains the codes of core IPMP related functionalities, including Tool List, Tool Container, and Rights Container handling, Tool Manager and Message Router. The output of this project is *MPEG2Splitter.lib* which is to be used by the following filter project

- *MPEG2SplitterFilter.dsp* in the same directory.

This project contains the codes of demultiplex filter. It will use the library functions in the above *MPEG2Splitter.lib.* The output of this project is *PSLMPEG2SplitterFilter.ax*.

Note, to launch the IPMP Terminal, the filter *PSLMPEG2SplitterFilter.ax* has to be registered in prior. Users can use Microsoft Visual Studio's Tools->Register Controls to register the filter.

- *PlayWnd.dsp* under subdirectory PSLPlayer.

This project contains the codes of IPMP Terminal. It is developed based on DirectShow. So users are required to install DirectX first to get the project build successfully. Users can go to Microsoft's website to get the latest DirectX developed package, which is free of charge. The output of this project is *playwnd.exe*.

Workspace Tsmuxer

The workspace *tsmuxer.dsw* contains 2 projects. They are,

- *PSL MPEG2 MessageInterface.dsp under subdirectory MessageInterface.*

This project contains the codes of IPMP Messages and the handling of the messages. They could be used by the multiplexer to generate IPMP message carried in the IPMP Stream. The output of this project is *PSLMPEG2MessageInterface.lib*.

Note, this project should be the same as the message project introduced above. Any changes to either project must be synchronised.

- *Tsmuxer.dsp* in the same directory.

This project contains the codes of reference multiplexer. The output of this project is a tsmuxer.exe.

- README and Sample configuration files under subdirectory Sample

A README file is provided as a user-manual to the Tsmuxer application. Also sample parameter files and sample descriptor data are provided to give users a start-up point.

Workspace DESDLL

The workspace *desdll.dsw* contains 2 projects. They are,

- *PSL MPEG2 MessageInterface.dsp* under subdirectory MessageInterface.

This project contains the codes of IPMP Messages and the handling of the messages. They are used by IPMP Tools, DES decryption tool in this case, to receive message from Terminal and generate messages to be sent back to Terminal. The output of this project is *PSLMPEG2MessageInterface.lib*.

Note, this project is the same as the message projects introduced above. Any changes to either project must be synchronised.

- *Desdll.dsp* in the same directory.

This project contains the codes of DES decryption tool. Besides the DES decryption functions, it has a Message Router as well to send and receive IPMP Messages. The interfaces between the tool and terminal are informative. The output of this project is *desdll.dll*.