ETSI
TECHNICAL
REPORT

ETR 225

October 1995

Source: ETSI TC-TE

Reference: DTR/TE-01063

ICS: 33.020

Key words: M&HIRS, multimedia platform, MHEG, API, script representation

Terminal Equipment (TE); Application Programming Interface (API) and script representation for MHEG; Requirements and framework

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Foreword

This ETSI Technical Report (ETR) has been produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI).

ETRs are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or the application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or an I-ETS.

Blank page

1 Scope

This ETR analyses the requirements for a standardized Application Programming Interface (API) and script representation and provides the framework for this future standardization work to be performed within ETSI. This ETR:

- defines a generic reference model describing the functional architecture of multimedia applications;
- details which new services and applications will benefit from the target ETSs;
- indicates why those services and applications require the development of these ETSs;
- investigates how those applications and services are likely to use the API and script representation;
- specifies the functions that should be provided by the API and script representation;
- identifies the technical requirements on the API and script representation; and
- evaluates technical options for the definition of the API and script representation, making recommendations accordingly and listing the identified open issues.

2 References

This ETR incorporates by dated or undated reference, provisions from other publications. These references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETR only when incorporated in it by amendment or revision. For undated references the latest edition of the application referred to applies.

[1] ITU-T Draft Recommendation T.170: "AVI - System: General Introduction, Principles, Concepts and Models".

[2] ISO/IEC DIS 13522-1: "Information Technology - Coding of Multimedia and Hypermedia Information".

[3] ITU-T Recommendation I.113 (1993): "Vocabulary of terms for broadband aspects of ISDN".

[4] ITU-T Recommendation I.112 (1993): "Vocabulary of terms for ISDNs".

[5] CCITT Recommendation Q.9 (1988): "Vocabulary of switching and signalling terms".

[6] ETR 173 (1995): "Terminal Equipment (TE); Functional model for multimedia applications".

[7] CCITT Recommendation X.208 (1988): "Specification of Abstract Syntax Notation One (ASN.1)".

[8] CCITT Recommendation X.209 (1988): "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".

[9] ISO/IEC 8879: "Standard Generalized Markup Language (SGML)".

[10] ETR 181: "Terminal Equipment (TE); Multimedia Portfolio; A compilation of multimedia applications and services provided by ETSI members".

[11] ETR 227: "Multimedia applications and services; Inband and outband signalling protocols; A survey".

[12] ETR 084 (1993): "Terminal Equipment (TE); Multimedia & Hypermedia Information Retrieval Services (M&HIRS), Investigation of candidate architectures for M&HIRS".

[13] ITU-T Recommendation I.374 (1993): "Framework Recommendation on ”network capabilities to support multimedia services””.

[14] Draft ITU-T Recommendation F.MDS (02/94): "Multimedia distribution services baseline document".

[15] ITU-T Recommendation I.211 (1993): "B-ISDN service aspects".

[16] ETR 228 (1995): "Terminal Equipment (TE); Broadband Multimedia Information Retrieval Service".

[17] ETR 176 (1995): "Terminal Equipment (TE); Interworking and interoperability of retrieval services and audiovisual services on narrow band networks".

[18] ITU-T Recommendation T.122 (1993): "Multipoint communication service for audiographics and audiovisual conferencing service definition".

[19] ISO/IEC 9646 Parts 1 to 5 (1991): "Information Technology - Open Systems Interconnection - Conformance testing methodology and framework".

[20] ETR 141 (1994): "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications; The Tree and Tabular Combined Notation (TTCN) style guide".

[21] IEEE Std 1224.2-1993: "Directory Services - API (Language Independent)".

[22] IEEE Std 1224-1993: "OSI abstract data manipulation - API (Language Independent)".

[23] IEEE Std 1326-1993: "Test methods for measuring conformance to OSI abstract data manipulation - API (Language Independent)".

[24] ISO/IEC CD 14478-1 (1994): "Presentation Environments for Multimedia Objects (PREMO)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this ETR, the following definitions apply:

application (when used alone): The same as telecommunication application (see below).

Application Programming Interface (API): A boundary across which application software uses facilities of programming languages to invoke services. These facilities may include procedures or operations, shared data objects and resolution of identifiers.

conversational service: An interactive service which provides for bi-directional communication by means of real-time (no store-and-forward) end-to-end information transfer from user to user. [ITU-T Recommendation I.113 [3], definition no. 114].

distribution service: Service characterized by the unidirectional flow of information from a given point in the network to other (multiple) locations. Distribution services are subdivided into two classes: distribution services without user individual presentation control and distribution services with user individual presentation control [ITU-T Recommendation I.113 [3], definition no. 119].

distribution service with individual presentation control: A distribution services in which the information is provided as a sequence of information entities, e.g. frames with cyclical repetition, so that

the user has the ability to select individual information entities and control the start and order of the presentation of the information [ITU-T Recommendation I.113 [3], definition no. 120].

function family: Cluster of functional MHEG API requirements that have the same target type (i.e. the kind of concept on which they apply) and the same type of operation that applies to this concept.

hypermedia: The ability to access monomedia and multimedia information by interaction with explicit links [ITU-T Draft Recommendation T.170 [1]].

interchange object: The interchangeable multimedia/hypermedia information, encoded according to ISO/IEC DIS 13522-1 [2], annex A.

interactive service: A service which provides the means for bi-directional exchange of information between users or between users and hosts. Interactive services are subdivided into three classes of services: conversational services, messaging services and retrieval services [ITU-T Recommendation I.113 [3]].

local application: A piece of software which is part of the (telecommunication) application and is running on the considered equipment.

mh-object: The presentation of an object within the MHEG engine, usuable for internal purposes (e.g. computation of link conditions).

MHEG application: A piece of software which uses the MHEG API. A MHEG application is therefore a client of an MHEG engine.

MHEG-using application: An application which involves the interchange of MHEG objects within itself or with another application (see MHEG).

Multimedia and Hypermedia (M&H) application: An application which involves the presentation of multimedia information to the user and the interactive navigation across this information by the user.

Multimedia and Hypermedia Information Retrieval Services (M&HIRS): A generic set of services which provide users with the capability to access and interchange multimedia and hypermedia information.

multimedia application: An application which involves the presentation of multimedia information to the user.

multimedia: The property of handling several types of representation media.

Multipoint Control Unit (MCU): A device that serves to connect terminals and other MCUs in a multipoint fashion. A MCU may include the functionality of a terminal as well as that of a MCU.

presentation object: A piece of information which is to be output to/input from the user.

primitive: Basic entry points provided by a provider module to any user module to enable the user module to access software services supplied by the provider module.

retrieval service: An interactive service which provides the capability of accessing information stored in database centres. The information will be sent to the user on demand only. The information can be retrieved on an individual basis, i.e., the time at which an information sequence is to start is under the control of the user [ITU-T Recommendation I.113 [3]].

rt-object: Copies of mh-objects usable for presentation to the user which are referred to as run-time objects.

Service Support Unit (SSU): A functional unit that provides the service functions for a given service.

service: That which is offered by an Administration to its customers in order to satisfy a specific telecommunication requirement [ITU-T Recommendation I.112 [4]].

software application: A piece of software answering a set of user's requirements and for use by a computer user.

telecommunication application: A set of a user's requirements [CCITT Recommendation Q.9 [5]].

terminal application: A piece of software running on the terminal and performing the part of the processing that is required to make the terminal appropriate for user access to the application. The terminal application is usually the "master" module in the terminal.

user: A person or machine delegated by a customer to use the services and/or facilities of a telecommunication network [ITU-T Recommendation I.112 [4]].

3.2 Abbreviations

For the purposes of this ETR, the following abbreviations apply:

3PTY	Three-Party supplementary service
ADSL	Asynchronous Digital Subscriber Line
ANDF	Architecture Neutral Distribution Format
AP	Access Point
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
AVCS	Audiovisual Conferencing Service
B-ISDN	Broadband Integrated Services Digital Network
CATV	Community Antenna Television
CBDS	Connectionless Broadband Data Service
CBR	Constant Bit Rate
CD	Compact Disk
CD-I	Compact Disk Interactive
CD-ROM	Compact Disc Read Only Memory
CONF	Conferencing supplementary service
CORBA	Common Object Request Broker Architecture
DAT	Digital Audio Tape
DIS	Draft International Standard
DQDB	Distributed Queue Dual Bus
DRA	Defence Research Agency
DSM CC	Digital Storage Media - Control Commands
DTD	Document Type Definition
DVB	Digital Video Broadcasting
EBNF	Extended Backus Naur Form
ECU	European Currency Unit
EPG	Electronic Programme Guide
EWOS	European Workshop for Open Systems
FDDI	Fibre Distributed Data Interface
FTTH	Fibre To The Home
GUI	Graphical User Interface
HDSL	Hybrid Digital Subscriber Line
ICS	Implementation Conformance Statement
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IMA	Interactive Multimedia Association
ISDN	Integrated Services Digital Network
IUT	Implementation Under Test
iVoD	Interactive Video on Demand
IWU	InterWorking Unit
JBIG	Joint Bi-level Image Experts Group
JPEG	Joint Photographic Experts Group
JTC	Joint Technical Committee
LAN	Local Area Network
M&H	Multimedia & Hypermedia
M&HIRS	Multimedia & Hypermedia Information Retrieval Services
MCU	Multipoint Communication Unit

MH	Multimedia hypermedia
MHEG	Multimedia and Hypermedia information coding Experts Group
MHEG-S	Multimedia and Hypermedia information coding Experts Group Script
MHI	Multimedia and Hypermedia Information
MPEG	Moving Picture Experts Group
N-ISDN	Narrowband Integrated Services Digital Network
OMG	Object Management Group
OSI	Open Systems Interconnection
PC	Personal Computer
PCI	Programming Communication Interface
PCO	Point of Control and Observation
PCTR	Protocol Conformance Test Report
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation Extra Information For Testing
PNO	Public Network Operator
PoI	Point Of Information
PON	Passive Optical Network
PoS	Point Of Sale
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
rt-objects	run-time objects
RTE	Run-Time Engine
SAF	Service Access Function
SGML	Standard Generalized Markup Language
SIR	Script Interchange Representation
SL	Scripting Language
SMDS	Switched Multimegabit Data Service
SSU	Service Support Unit
SUT	System Under Test
sVoD	Staggered Video On Demand
TE	Terminal Equipment
TTCN	Tree And Tabular Combined Notation
TV	Television
UTP	Unshielded Twisted Pair
VAP	Videotex Access Point
VCR	Video Cassette Recorder
VHS	Video Home System
VoD	Video on Demand
WAN	Wide Area Network

4 General model

This clause defines a generic reference model describing a functional architecture common to the target multimedia applications. The model is a refinement of ETR 173 [6] for an MHEG environment. The model is applicable to all applications that interchange or process Multimedia and Hypermedia (M&H) objects encoded according to the MHEG standard. The reference model may be applicable in some parts as well if other coding standards different from MHEG are used but such an investigation is outside the scope of this ETR.

Figures 1, 2 and 3 show the reference model for terminal-to-host, terminal-to-terminal and terminal-to-database configurations. Together they form a generic model which is valid for the following three different configurations for both point-to-point and multipoint communication:

- terminal-to-host;
- terminal-to-terminal;
- terminal-to-database.

For a "terminal-to-host and database" configuration the host may use the end-to-end protocol between access agents to reference objects or contents at the database. The database has the same structure as shown in figure 3.

For a terminal-to-host architecture, the Service Support Unit (SSU) to which the terminal may be connected may also enable the user to select between different applications. The host is connected to the SSU via the host access network.

For a terminal-to-terminal architecture the SSU to which all terminals are connected may be a Multipoint Control Unit (MCU) that controls and manages the application and the different terminals. The end-to-end protocol between an application and a distant MHEG engine is applicable to all terminals. Each terminal application can use the protocol to communicate with each other terminal engine.

For a "terminal-to-database" or "terminal-to-host and database" structure the database may mainly consist of an access agent used to locate the referenced objects.

The reference model is valid for all categories of services (retrieval, distribution, conversational) but the actual implementation of the application architecture may be different.

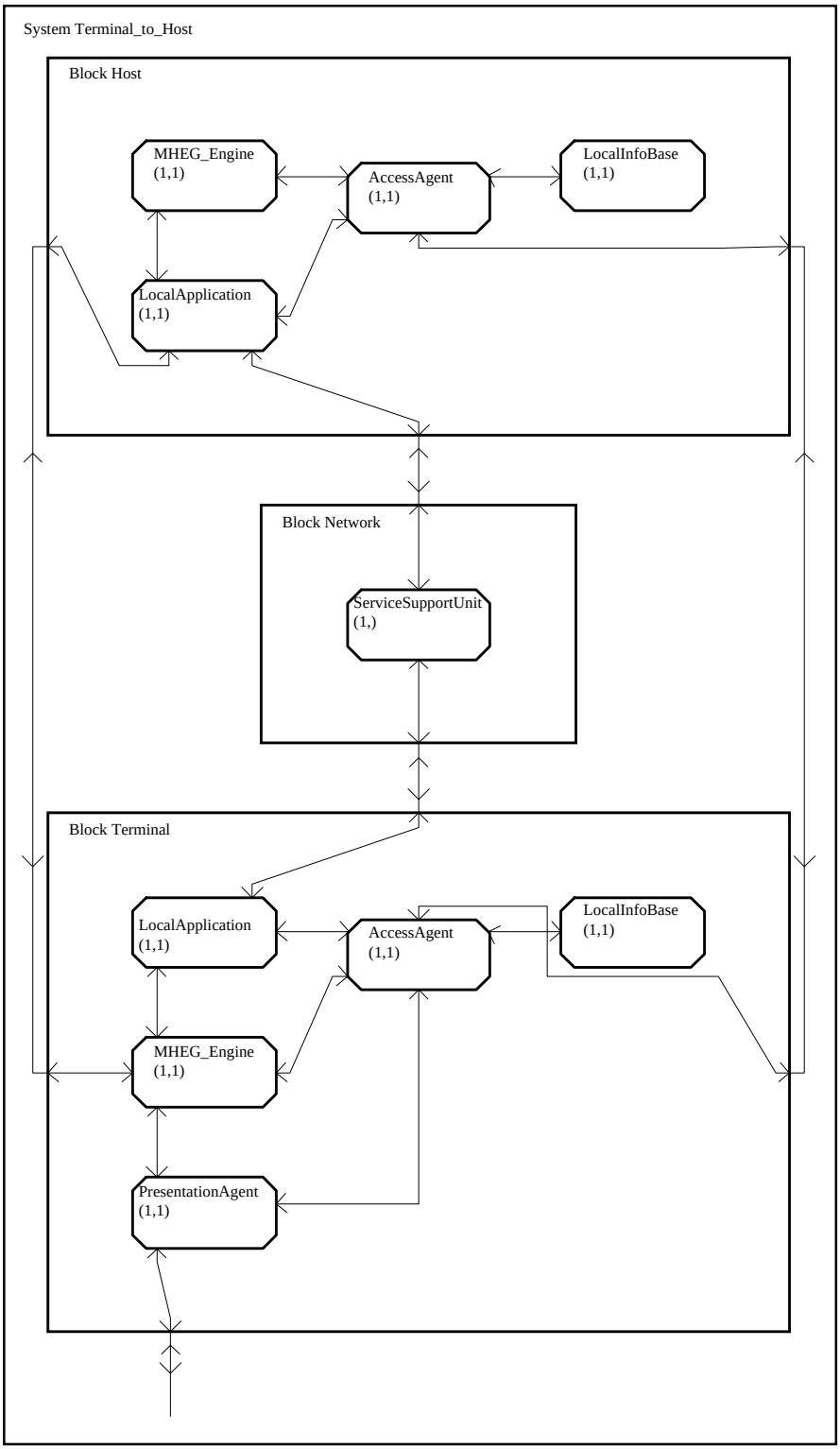

Figure 1: Application architecture reference model for terminal-to-host configurations

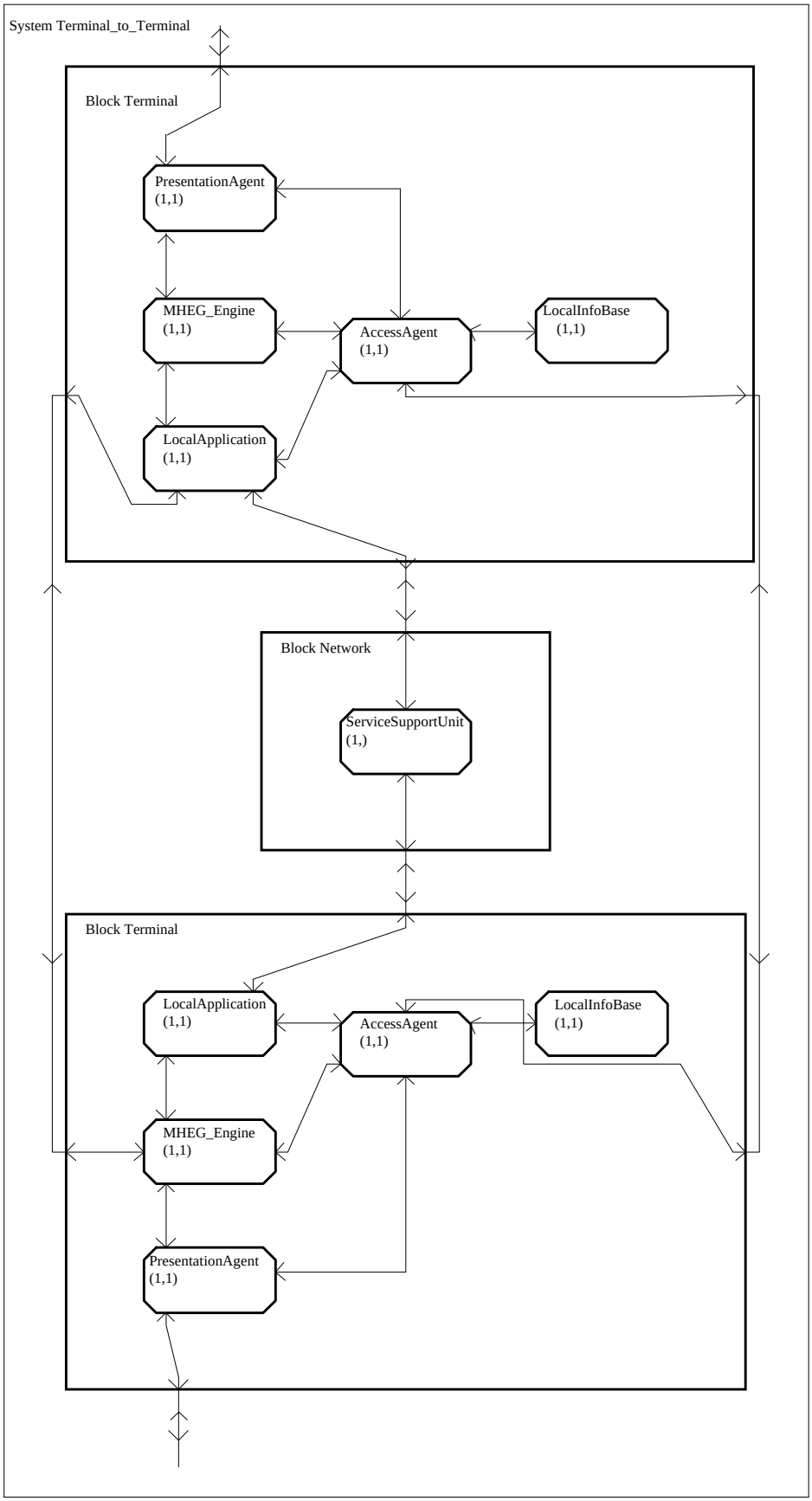

Figure 2: Application architecture reference model for terminal-to-terminal configurations

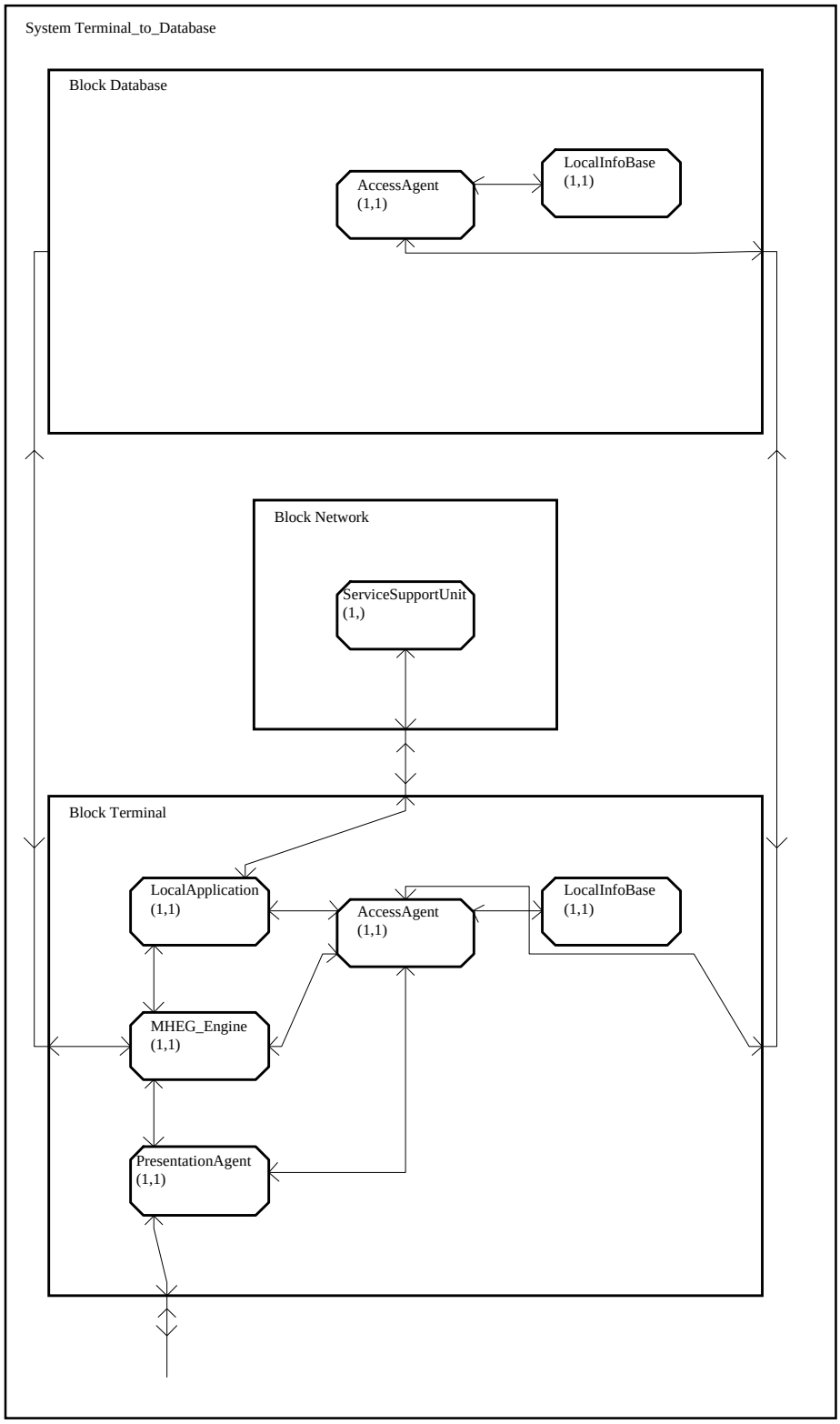

Figure 3: Application architecture reference model for terminal-to-database configurations

In a M&H architecture the following functional units can be identified:

- presentation agent;
- access agent;
- local application interpreter;
- MHEG engine;
- local information base.

In a M&H architecture the following Interfaces (APIs) can be identified:

a) the services of the presentation agent are offered through the presentation API;
b) the services of the MHEG engine are offered through the MHEG API;
c) the services of the access agent are offered through the access API.

In a M&H architecture the following end to end protocols can be identified:

1) end-to-end protocol between an application - distant MHEG engine;
2) end-to-end protocol between an application - service support function;
3) end-to-end protocol between an access agent - access agent.

The **presentation agent** manages the presentation of monomedia data, performs data format decoding and manages the user interaction. It also acts as an interface to external devices like smart card readers, Video Cassette Recorder (VCRs), etc,... The presentation agent is accessed by its clients (here the MHEG engine) through the "presentation API" which isolates the software on higher level from the specific features of the various hardware sub-platforms; the main components of the presentation agent are Joint Photographic Experts Group (JPEG) decoders, Moving Pictures Experts Group (MPEG) decoders, Videotex syntax decoders as well as decoders from input devices (keyboard, mouse, voice recognition, etc,...). The contents to be presented following a request through the presentation API are normally specified by reference identification; the reference is input to the access agent. The presentation agent is only present on a terminal.

The **access agent** manages a directory that enables it to locate the requested content. The content is either on the local object base because it is resident or has been previously downloaded, or it is located on a distant memory (distant multimedia database or cache memory in the network); in the latter case, the access agent automatically sends the appropriate request through the network interface to get the object or the content. The access agent makes the location of the various objects (multimedia contents, MHEG objects, scripts, application specific data) totally transparent to its clients, i.e. the processes trying to access the objects.

The **local application** manages the logic of a given platform. The application itself will often be distributed between several platforms (terminals, hosts). The local application is a client of the access agent via the access API, of the MHEG engine via the MHEG API and of the presentation agent via the presentation API. The local application may include in some cases a script processor allowing to execute scripts, i.e. parts of the application which are interchanged during the course of an application. The script processor is a functional element in charge of executing scripts. It may also be the script itself (if interchanged in executable form), a script language interpreter or a Script Interchange Representation (SIR) interpreter.

The **MHEG engine** interprets MHEG objects, manages links between them, triggers actions and orders objects presentation and access. It is controlled by the application through the MHEG API. The MHEG services accessible through the MHEG API may be used by any application based on the MHEG standard to exchange MHEG objects. Just as the presentation agent, the MHEG engine uses the access agent each time it has to process a referenced MHEG object.

The **local information base** may be used to store objects, contents, scripts, etc,... permanently or temporarily on the device. No assumption is made on how those objects are stored and which physical storage device is used.

The **presentation API** allows the terminal MHEG engine to access the services offered by the presentation agent. The services may include primitives like "display data", "accept user input", etc,... The presentation API is beyond the scope of this ETR.

The **MHEG API** allows applications to access the services of the MHEG engine. The application may run locally in the application interpreter or on an remote device. If the application is running on an remote device it needs to access the API MHEG using the end-to-end protocol "application - distant MHEG engine". The services may include primitives like "prepare object", "destroy object", etc,... The functional and technical requirements for the MHEG API are specified in this ETR. The specification of the MHEG API is to be part of the scope of the standard to be prepared by ETSI.

The **access API** allows the clients of the access agent (local application, script interpreter, MHEG engine, presentation agent) to access the services offered by the access agent. The services may include primitives like "get object", "send object", etc,... The access API is beyond the scope of this ETR.

The **end-to-end protocol "application - distant MHEG engine"** allows an application running on a distant device to communicate with a local MHEG engine. This protocol enables the implementation of host - terminal configurations. The end-to-end protocol "application - distant MHEG engine" is beyond the scope of this ETR.

The **Service Support Unit (SSU)** is a functional unit that handles service specific control parameters and offers functionality that depends on the particular service. An example for a service support unit is the MCU in the case of the Videoconferencing service. The functionality handled by the SSU is service specific and not further described in this ETR.

The **end-to-end protocol "application - service support unit"** enables the use of the services provided by the service provider by the terminal application or the user respectively. The end-to-end protocol "application - service support unit" is beyond the scope of this ETR.

The **end-to-end protocol "access agent - access agent"** enables the handling, maintenance and exchange of objects and data in an distributed environment. The end-to-end protocol "access agent - access agent" is beyond the scope of this ETR.

5 An introduction to the MHEG standard

The following clause presents a general introduction to the MHEG standard. It is derived from the introductory clause of ISO/IEC DIS 13522-1 [2] ("Information Technology - Coding of Multimedia and Hypermedia Information").

5.1 Multimedia/Hypermedia application requirements

In the various domains, interactive multimedia systems are perceived to have increased impact on users through their use of image, video and audio in addition to traditional text information.

The development of these systems is based upon support for multimedia information in the following areas:

- **platforms:** emerging compression standards, such as JPEG or MPEG have enabled the development of dedicated hardware, which in the future will be provided as a standard system resource;

- **storage:** the increasing capacity of magnetic and optical disks combined with compression techniques enables the storage of increasing amounts of high bandwidth information;

- **communication:** the increasing availability and bandwidth of digital transmission media such as ISDN and other broadband telecommunication and broadcasting networks enable the appropriate exchange of large amounts of data.

Using these facilities, it is anticipated that multimedia applications will be designed to run on heterogeneous platforms and will be interconnected to offer multimedia services. For example, such applications may include computer supported multimedia co-operative work, multimedia messaging, electronic publishing and electronic books, audiovisual telematic systems for training and education, simulation and games, sales and advertising, home shopping, interactive television, Video on Demand (VoD) applications, and entirely new classes of multimedia applications.

These multimedia applications and services will use large quantities of structured multimedia objects resident in workstations, stored on digital interchange media, retrieved or distributed from remote sources through a network. As production of these multimedia data represents a significant investment, it is vital that it remains applicable in a world of rapidly evolving systems and technologies. In particular, information should be interchangeable between the data structures supported by different applications.

MHEG-using applications involve the following application requirements:

- multimedia database retrieval;
- frequent updates of multimedia data;
- manipulation of a set of data elements;
- creation of multimedia documents on a range of workstations;
- composition of multimedia data in time and space;
- association of a pan of a document with some other part of the same or another document (e.g. hypertext);
- synchronisation with types as follows:

    a) elementary synchronisation - two objects are synchronised either both with regard to the same reference origin time (parallel mode), or one with regard to the other (sequential mode);

    b) chained synchronisation - a set of objects is to be presented one after another in the form of a chain;

    c) cyclic synchronisation - one or more objects will be repetitively presented;

    d) conditional synchronisation - the presentation of an object is linked to the satisfaction of a condition.

- re-use of multimedia data by integration in different documents;
- exchange of multimedia data among heterogeneous systems;
- wide range of users, including closed user groups;
- real-time interactivity including acquisition of multimedia data;
- telecommunication of multimedia data;
- broadcast of multimedia data;
- wide range of data volumes and transfer rates;
- use of a wide range of terminals and workstations, including devices with minimal resources;
- access control, security;
- tariffing;
- copyright, licensing;
- wide range of support materials;
- minimal resources - facilities that the interchange form should provide for a given set of objects running in limited resources environments to meet their functional specification as determined by its designer, as follows:

    1) ease of specification of the minimum recommended resources needed by the MHEG application;

    2) ease of application - minimum requirements of an application should be easily recognised by the negotiation process;

    3) flexibility - the mechanism should be reliable across a broad range of system configurations and media formats;

    4) informative rather than restrictive - the using application should have the final determination as to whether a given set of objects is interpretable or not;

    5) extensible open - the representation should function correctly as new formats, classes, and application dependent parameters are added;

    6) scalability - trade off among different application platform capabilities, for example, resolution enhancement relative to presentation time;

    7) compositionality - the minimum resource requirements for a given MHEG application should be combinable in some consistent and predictable manner when applications are combined;

    8) graceful degradation - the primary concern of minimal resource systems is the manner in which degradation can be handled. An application should be able to express its policy with regards to this issue, for example by using scalability and related mechanisms.

- real-time interchange and presentation, as follows:

    - object placement optimisation - objects which are likely to be accessed simultaneously are adjacent from the standpoint of the access mechanism;

    - progressive access of objects - images may be retrieved and presented in increasing resolution for systems with significant presentation delay. Scalable versions of objects may be represented and retrieved for systems with insufficient resources for full fidelity presentation;

    - partial object retrieval - large objects may be retrieved in several portions since the entire content will not be presented at one time;

    - object sequencing - the order in which objects are expected to be presented should be used by the access mechanism when insufficient throughput would lead to unacceptable delays for the whole object to be interchanged;

    - separate retrieval of object description and object content - the object description should be retrieved without necessarily retrieving the content so that the system can use information about a set of objects to optimise the access for this set and resources needed for the access can be prepared;

    - global object index - a table of all objects and their position in an object set should be provided to support fast lookup of objects;

    - object interleaving - objects which are to be retrieved simultaneously may be interleaved so that large objects do not cause delays for other objects;

    - resource requirements for retrieval and presentation by the target system should be available by lookup rather than by derivation;

    - aggregate retrieval - objects may be aggregated into collections so as to be retrieved by one step rather than through a series of request.

These requirements can be summarised in a set of generic needs:

- application designers and distributors may require application portability in a multi-vendor environment;

- applications may need to handle multimedia information structured in such a way that real-time interactivity (including acquisition of multimedia data), as well as real-time interchange of multimedia data can be ensured;

- applications may need to compose and synchronise multimedia data in space and time;

- applications may need to be able to define links between multimedia data elements;

- applications may need to be able to reuse multimedia data by integration in different contexts;

- applications may need to be able to frequently update the multimedia data, as well as to manipulate a set of data elements;

- applications may need to be able to be interpreted on different systems, from minimal resources to non-limited resources systems;

- applications may need to be interchanged and presented in real-time.

5.2 Rationale for standardization of Multimedia and Hypermedia information

The rationale for a standard defining M&H information structures is based on the following considerations:

- standardization only at the level of monomedia information is not sufficient to guarantee application portability. Applications do not use the monomedia data separately but need to define an associated set of parameters which are necessary for presentation (such parameters may include identification of the encoding algorithm applied to the data, decoding parameters relevant to the data, optional attributes to be used for presentation, etc,...);

- standardization only at the level of monomedia information is not sufficient for the design and interchange of M&H information. M&H applications rely on abstractions or data structures which provide features such as synchronisation links and logical links between monomedia data;

- the design and management of M&H applications in distributed environments will be eased if the internal details of the information presentation are masked from the application by the use of appropriate abstractions. The application should only deal with functions such as management of information distribution, scheduling of presentation, management of the user dialogue and other high level activities.

The MHEG standard aims to provide generic Multimedia/Hypermedia information structures which fulfil these requirements and additionally are suited to:

- real-time presentation using multimedia presentation and synchronisation facilities available on the application platform;

- real-time interchange using communication facilities available on the application platform;

- final form representation in which the information is represented and coded for direct presentation, without requiring additional processing of its structure.

5.3 The MHEG standard objectives

5.3.1 MHEG for interchange

The MHEG standard is intended to provide interchange facilities for various media types. The media data may be encoded according to other international standards, e.g. JPEG for still image, MPEG for video, or may be encapsulated using private and proprietary coding techniques. The interchange units defined by MHEG are handled by the MHEG system under the control of an application.

In order to support multimedia interchange (as opposed to the interchange of multiple media), the MHEG standard provides structures for the composition of different media types within a single unit of interchange.

5.3.2 MHEG for presentation

The MHEG standard is intended to support final form presentation of multiple media types. The standard provides facilities for the identification of the coding technique to enable the use of the appropriate presentation resources on a specific platform.

In order to support multimedia presentation, as opposed to the presentation of multiple media, the MHEG standard provides structures for the composition of different media types in a presentation. This composition takes the form of time sequencing, spatial positioning and logical interaction between the media.

In addition, the MHEG standard supports interaction with, and modification of, media data and its associated presentation attributes, e.g. size, position, volume.

5.3.3 MHEG and minimal resources

The MHEG standard is intended to support the specification of the minimal resources required to present the encoded data. The MHEG standard provides facilities for the interchange of information relating to these resources which is provided by the information source.

5.3.4 MHEG for real-time

The MHEG standard is intended to ease the real-time interchange of multimedia information. The MHEG standard provides facilities to assist a using system in achieving an adequate information flow.

5.4 MHEG concepts

This subclause provides a general introduction to the MHEG concepts. The following concepts are presented:

- **MHEG class, MHEG objects:** The MHEG standard defines object classes. From these classes, MHEG objects may be instantiated by the object designer and interchanged between applications. Any number of MHEG objects may be instantiated from a given MHEG class;

- **run-time objects (rt-objects):** The MHEG model class is an MHEG abstract class which provides the following MHEG classes: script, content, multiplexed content and composite classes. From these model classes, model objects may be instantiated by the object designer and interchanged. In order to reuse the data interchanged in the model objects in different contexts, the object designer is able to create rt-objects from a given model object; for instance, the same data in a content object may be presented twice at the same time with different sizes using two rt-content objects but the content object will be interchanged only once. Any number of rt-objects may be created from a given model object by the object designer;

- **channels:** The MHEG standard defines logical spaces in which the rt-objects are presented and perceived by the user.

5.4.1 MHEG classes

The MHEG standard defines a classification of structures corresponding to units of multimedia/hypermedia information to be interchanged. This classification is based on an analysis of the common behaviour and properties of Multimedia/Hypermedia information and takes the form of an object-oriented approach to the standardization. This approach results in autonomous and reusable information structures which are generic to Multimedia/Hypermedia applications.

The MHEG classes contain a level of complexity that is compatible with their use as generic structures in a wide range of applications and domains. It is assumed that the semantics associated with the use of MHEG classes is defined at the application level and not at the MHEG level.

It should be understood that the use of an object-oriented approach to the definition of MHEG structures does not imply that a MHEG system should be based upon this approach. As the MHEG standard defines an interchange format for multimedia/hypermedia information, it makes no assumptions about the internal representation of MHEG structures or the design of MHEG systems, engines, interpreters, tools or using applications.

The MHEG standard provides a coded representation of each MHEG class. Instances of these coded representations are MHEG objects and represent the data to be interchanged and presented by Multimedia/Hypermedia applications.

The base-coded representation makes use of CCITT Recommendation X.208 [7] and CCITT Recommendation X.209 [8] and is described in ISO/IEC DIS 13522-1 [2].

An alternate coded representation is provided in the form of a Document Type Definition (DTD) as defined by ISO/IEC 8879 [9]. This coded representation is isomorphic to the base coded representation and is described in ISO/IEC DIS 13522-1 [2].

These two notations and encodings Abstract Syntax Notation One (ASN.1) and Standard generalized Markup Language (SGML) are alternatives used to express the same and unique representation of the MHEG objects.

5.4.2 Overview of the MHEG classes

The classes defined in the MHEG standard (ISO/IEC DIS 13522-1 [2]) can be used to specify:

- objects containing media information;
- relationships between objects;
- dynamic behaviour of objects;
- information to optimise the real-time handling of objects.

The following instanciable classes are defined in ISO/IEC DIS 13522-1 [2].

5.4.2.1 Content class

The content class is a model class. It contains, or refers to, the coded representation of media information together with a parameter set containing information required for content presentation. This parameter set contains an identification of the coding method and a field for the specification of application-oriented parameters (e.g. fonts, colour table). The content class specifies also the original size, duration and volume of the data. These values are expressed using generic space units.

5.4.2.2 Multiplexed content class

The multiplexed content class is a model class which is a sub-class of the content class. It contains, or refers to, the coded representation of a multiplexed media data together with a description of each multiplexed stream.

5.4.2.3 Composite class

The composite class is a model class. It provides the support for associating M&H objects with each other. This mechanism provides a consistent approach to the synchronisation in time and space and linking of a set of objects. This class provides also the logical structure to describe the list of possible interactions offered to the user, but does not define the interaction facilities provided by the user interface. Such interaction may be achieved in a variety of ways for example: graphical user interface, keyboards, etc,... The MHEG standard does not define the "look and feel" of multimedia interactive presentations, neither does it propose to change or add concepts to those existing in typical graphical user interface. As the MHEG standard is generic and independent of platform and implementation, it describes interaction at a virtual level. It is up to a using application to apply these mechanisms using its specific "look and feel".

5.4.2.4 Action class

The MHEG standard provides an initial behaviour for each MHEG object, rt-object and channel, e.g. default position of a rt-content. The MHEG standard also provides a means to modify the initial behaviour of each object by defining a list of elementary actions to be applied on the objects. The modification of the behaviour is achieved by interchanging the corresponding elementary actions within action objects. The action objects are used within a link object to describe the link effect.

The action class defines a structure which specifies a synchronised set of elementary actions to be applied on one or more targets.

The MHEG standard defines elementary actions whose application may affect the following behaviours of an MHEG object, an rt-object or a channel:

- **preparation:** actions are provided to control the availability of the MHEG object in the system. For example, "Prepare" and "Destroy" actions may be applied to add and remove an MHEG object from the system;

- **creation of rt-objects:** actions are provided to create rt-objects from a model object;

- **presentation:** actions are provided to control the progress of the rt-components in the system. For example, "Run" and "Stop" actions may be applied to control the progression of a time-based rt-component;

- **rendition:** actions are provided to control the rendition of the rt-component on the system. These actions vary according to the media type, for example, "Set Speed" for time-based media and "Set Size" for visible media;

- **interaction:** actions are provided to control the results of interaction with a rt-component in the system. For example, "Set Selectable" and "Set Modifiable" specifies respectively the selectability and the modifiability of a rt-component;

- **activation:** actions are provided to control the activation of the rt-scripts in the system.

The MHEG standard also defines a means to retrieve the behaviour of an MHEG object, an rt-object or a channel:

- **evaluated value:** actions are provided to get the behaviour attribute or status value of MHEG objects, rt-objects and channels. These actions are used to express the link condition in a link object.

5.4.2.5 Link class

The link class defines a structure which specifies a set of relationships. Each relationship is defined between one or more "sources" and one or more "targets". The relationship is composed of conditions associated with the sources (link condition) and actions to be applied to the targets (link effect). The actions which are described in action objects are to be applied on the targets when the conditions are satisfied.

The source and target can be instances of any MHEG class, including link class and action class instances. The source and targets can also be any rt-objects.

Instances of the link class are used to specify the time sequencing, spatial positioning or logical interaction between MHEG objects and rt-objects.

5.4.2.6 Script class

The script class is a model class. It defines a container for **complex** relationships between MHEG objects and rt-objects, defined by a non-MHEG language. The MHEG standard does not define the scripting language itself, but provides the script class to encapsulate a script and an indication of the language used.

The evaluation of the requirements for a scripting language or a SIR is within the scope of this ETR.

It is assumed that the scripting language used in a script object is able to reference MHEG objects, rt-objects and to access their attributes.

5.4.2.7 Descriptor class

The descriptor class defines a structure for the interchange of resource information about a single or a set of other interchanged objects. The described objects are called related objects. The information can be used to facilitate a correspondence between the resources required to present the objects and the resources available to the system, or to perform a negotiation between the source of the MHEG objects and the presentation site.

5.4.2.8 Container class

The container class provides a container for regrouping M&H data into a single interchange unit.

5.4.3 Run-time objects (called rt-objects)

For the purpose of reusing model objects (script, content, multiplexed content and composite objects) in different presentations or activations, a clear separation is made between the interchanged **model** object, which contains the reusable data or composition, and the **rt-object** corresponding to a specific view of the “model” data or composition.

The rt-objects (rt-script, rt-content, rt-multiplexed content and rt-composite objects) are created by the object designer using the MHEG "new" action. The presentation or activation of a rt-object does not affect the model object, this allows the reuse of a same model object in different rt-objects.

5.4.4 Channels

A channel is a logical space in which the rt-components (rt-content, rt-multiplexed content and rt-composite objects) are presented and perceived by the user. The channels are created by the object designer using an MHEG action. The object designer provides the desired mapping of the channels within the descriptor object. The channels are mapped to the real world by the MHEG engine.

5.5 The MHEG API

The MHEG standard does **not** define an API for the handling of objects in a system. This will be defined as part of the further work within TC-TE.

5.6 Extendibility of the MHEG standard

The scope of the classes defined in the MHEG standard is compatible with their use in a wide range of applications and domains. It is recognised that certain applications may require specific functionality not directly provided by the classes. For example, an application may require the use of specialised resources available on a given platform. In this case, additional resources which are not provided directly by the MHEG standard may be associated to the MHEG objects. This association is not defined by the MHEG standard but may be defined by other standards or applications using one of the following techniques. These techniques do not change the coded representation of the facilities defined by the MHEG standard.

a) A using application may define new additional elementary actions to be targeted to MHEG objects or rt-objects. The new elementary actions do not modify the MHEG action class.

   NOTE 1: The new elementary actions are ignored by "basic" MHEG engines which support only the facilities defined by the MHEG standard.

b) A using application may define new additional attributes of the objects. Two extensions may be provided by the using application without changing the MHEG object representation:

   1) a using application creates its own classes by derivation of the MHEG classes. These classes are considered as using application classes and are no longer MHEG classes. A "basic" MHEG engine does not recognise these new classes;

   2) additional elementary actions may be provided by the using application as in a) in order to set a new attribute to an MHEG object target. A "basic" MHEG engine ignores the action, an extended one takes them into account, and thereby introduces the new attribute.

c) A using application may define new additional media types, the MHEG standard allows the definition of proprietary media types which are provided by the using application.

   NOTE 2: For example, a using application may create and manipulate new attributes such as payment or colour or it may vary some speech modulation parameter in an audio object.

6 Service requirements

6.1 Main categories of services

The applications that interchange M&H information can be classified in to three broad families allowing combinations between them. The three families are:

- the retrieval family;
- the distribution family;
- the conversational family.

   NOTE: For the studies carried out in this ETR it was found more appropriate to describe the VoD service together with the services of the distribution family, although it clearly belongs to the retrieval family.

This subclause investigates each different family and its special functional requirements on the MHEG API and on the SIR. The goal is that the API and the SIR that will be the result of the requirements evaluated in this ETR is applicable to the broadest possible range of M&H applications.

Each family of application is subdivided into different categories that are described in terms of architecture, information transfer scenario, specific TE, etc,... Each category is developed on the basis of an example of "pilot" application selected according to the following criteria:

- interest received up to now from ETSI (ETR 181 [10] and ETR 227 [11]);

- interest publicly claimed up to now from Public Network Operators (PNOs) and commercial application providers, according to the expected market.

The summary of the specific requirements of all presented applications will then form the basis for general functional requirements of M&H applications on the API and SIR.

6.2 Applications of retrieval services except VoD

6.2.1 Common characteristics and taxonomy

In all retrieval applications, the user employs a terminal to communicate with a M&HIRS. The application may either run on the users TE or on a remote host or may be distributed between several different entities. Objects which contain data belonging to an application may be stored in a local database, in one or several remote databases, or in a distributed database they may consist of local and remote databases as well. ETR 084 [12] gives further details of M&HIRS candidate architectures.

Following the multimedia portfolio that was launched by TC-TE,

- most M&HIRS will be mainly used by professional/enterprise users, but some will be dedicated to residential users;

- a number of used terminals will not be dedicated to M&HIRS but will also have facsimile, messaging service and Videotex access capabilities.

Important service/application parameters are:

- establishment of the communication on demand;
- establishment of the communication around the clock;
- user identification;
- access/control priority right management;
- local storage of information;
- local processing of information;
- all the user terminals are not on the same network;
- use of several services/media simultaneously;
- variable allocation of bandwidth for the media;
- authentication/signature;
- copyright protection/copy control;
- charging;
- heterogeneous multimedia platforms;
- all contents shared by all terminals.

6.2.1.1 Networks

According to the multimedia portfolio, some user terminals are likely to connect to retrieval services via the Public Switched Telephone Network (PSTN). Integrated Services Digital Network (ISDN) or Broadband ISDN (B-ISDN) are, however, the most widely envisaged. Other target networks include Local Area Network (LAN), Wide Area Network (WAN), Community Antenna Television (CATV) and Internet. For most services, the transmission rate requirements should not exceed 2 Mbit/s. ITU-T Recommendation I.374 [13] gives further information on 64 kbit/s and B-ISDN expected network capabilities.

6.2.1.2 Terminal Equipment

As half of the M&HIRS will be dedicated to professional/enterprise users, the TE is likely to be PC-based. For specific applications (tourist information, etc,...) public terminals are best suited. Residential users may access M&HIRS using multi-functional dedicated terminals.

6.2.1.3 Service attributes

The following service attributes characterise applications based on retrieval services:

- terminal-to-host, or terminal-to-database architecture;

- bi-directional asymmetric communication;
- information is supplied on demand (high level of interactivity);
- most often point-to-point connections.

M&HIRS involve a wide variety of possible information and control exchange scenarios. In a given application, M&H objects can be stored either in a remote database or locally in the terminal. A scenario where objects are distributed, i.e. can be stored in both places as well.

M&H applications may be available on the terminal as well as on a host. Applications that are distributed between the terminals and the host may also be available. Taking into account the previous considerations, M&HIRS can be separated into three different categories where every combination of these main categories can occur. The categories are:

a) application running on the terminal retrieving M&H objects from a remote database (this is developed in subclause 6.2.2);

b) distributed application between terminal and host exchanging M&H objects (this is developed in subclause 6.2.3);

c) applications running on the host sending M&H objects to the terminal (this is developed in subclause 6.2.4).

6.2.2 Multimedia retrieval services featuring local application and remote data

This category comprises all scenarios where the application logic is on the terminal device operated by the user and M&H data is stored in the network or on an M&H database connected to the network.

6.2.2.1 Application example: encyclopaedic applications, electronic libraries and electronic books

The following features characterise encyclopaedic applications, electronic libraries and electronic books from a user's perspective. A user:

- consults a core encyclopaedia on CD-ROM, updates (subscription) through regular downloads, hot topics available on-line;

- consults yellow pages, tourist guides using electronic publishing of MHI material.

The following application parameters are required:

- user identification required;
- local storage of information;
- authentication/signature;
- copyright protection/copy control;
- charging.

6.2.2.2 Service attributes

The following service attributes characterise this category of applications:

- terminal-to-database configuration;
- bi-directional asymmetric communication;
- establishment of the communication around the clock;
- establishment of the communication on demand;
- information is supplied on demand;
- point-to-point connection.

6.2.2.3 Networks

According to the complexity of data all networks mentioned in subclause 6.2.1.1 are possible candidates starting with the PSTN at a transmission speed of 9,6 kbit/s. If the application design demands frequent retrieval of M&H objects, networks with a higher data throughput (such as ISDN) are preferable.

6.2.2.4 Terminal Equipment

The TE for this application is likely to be PC-based.

6.2.2.5 Information interchange scenario

Most of the MHEG objects are stored on a local database at the user's site. Hot topics are stored in the M&H database and are transmitted to the terminal on user request. Updates of the locally stored MHEG objects are available at the database as well and are transmitted to the terminal on the user's request. The connection establishment will, in most cases, be handled automatically by the local application. The services offered by the SSU will be used to connect the terminal to the desired database.

6.2.2.6 Application configuration

The terminal is owned by the user. The SSU is operated by the service provider and responsible for connecting the terminal to the desired M&H database. The M&H database is operated by the information provider.

Figure 4 below shows the configuration of encyclopaedic applications, electronic libraries and electronic books.

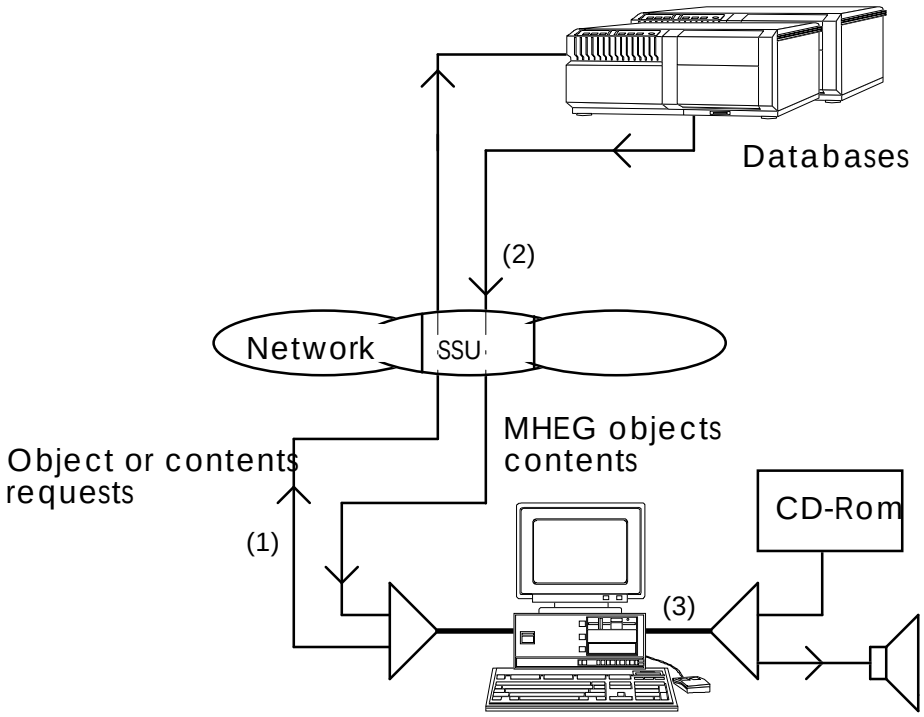

Figure 4: Specific configuration of encyclopaedic applications, electronic libraries and electronic books

The following communication channels are used in the example figure above:

(1) MHEG object, or content request from terminal to database;
(2) MHEG objects and data content from database to terminal;
(3) external input and output devices connected to the terminal.

Application requests for objects that are not present at the terminal result in network requests for the corresponding objects. The access agent in the terminal provides a data directory function and locates the requested data.

6.2.2.7 Use of MHEG

Much material that is currently published on paper could be enhanced by the integration of audiovisuals and by being available possibly in a hypertext manner over a telecommunication network. For example,

tourist guides and yellow pages could be a mixed sales and advertising application. MHEG is an appropriate standard to support such hypertext architectures in a distributed environment.

MHEG objects are stored mainly on the storage devices at the terminal. Objects that are requested from a remote source contain additional new information. The access agent in the database is able to interpret the MHEG object reference correctly and can use this information to locate the objects requested from the terminal at the different physical databases. The information provider generates his information by means of a MHEG object generator and stores it on his database which is connected to the SSU.

6.2.2.8 Use of scripts

In this application example scripts are most likely to be used for the updating of the terminal application. This allows the application provider to provide a large number of users simultaneously with a new application release in a very convenient way.

6.2.2.9 Application architecture

The application architecture corresponds to the terminal-to-database configuration presented in the general model.

6.2.2.10 API requirements

Table 1 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 1

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	
Handling	✓
Behaviour	✓
Exception	✓

6.2.2.11 SIR requirements

Table 2 gives an overview of the SIR functionalities that are required in this particular application.

Table 2

Function family	Needed
external device control	✓
external device control for data acquisition	
manipulation of MHEG objects	✓
access to external data	✓
access to external advanced calculation capability	
computations, variable handling and control structures	✓

6.2.3 Multimedia retrieval services featuring distributed application and distributed data

This category comprises all scenarios where the application logic and the M&H data are distributed between the terminal device and a host with a possibly associated M&H database.

6.2.3.1 Application example: Point of information, Point of sales

The following features characterise point of information and point of sales applications from a user's perspective. A user:

- consults information booths with a city guide including street maps and orientation help. The information is regularly updated (traffic situation, cinema and restaurant programmes, special events...) from a remote source and may be combined with advertising (ITU-T Recommendation F.740 gives similar examples);
- does tele-transactions (ticket and other reservations);
- does telebanking transactions (money orders);
- does teleshopping.

The following application parameters are required:

- local storage of information;
- local processing of information;
- authentication/signature;
- access/control priority right management;
- copyright protection/copy control;
- charging;
- all the user terminals are not on the same network;
- use of several services/media simultaneously;
- variable allocation of bandwidth for the media;
- heterogeneous multimedia platforms.

6.2.3.2 Service attributes

The following service attributes characterise this category of applications:

- terminal-to-host configuration;
- bi-directional asymmetric communication;
- establishment of the communication around the clock;
- establishment of the communication on demand;
- information is supplied on demand;
- point-to-point connection.

6.2.3.3 Networks

According to the complexity of data, all networks mentioned in subclause 6.2.1.1 are possible candidates starting with PSTN at a transmission speed of 9,6 kbit/s. If the application design requires frequent retrieval of M&H objects, networks with a higher data throughput (ISDN) are preferable.

6.2.3.4 Terminal Equipment

Public or private information and transaction terminals will form a significant part of the used TE. Dedicated terminals for residential users and PC-based terminals are likely to be used as well.

6.2.3.5 Information interchange scenario

MHEG objects are stored on a local database at the user's site as well as on M&H databases connected to the host. According to specific applications or sub application parts of a given application these objects are interchanged among the involved units. There is also a need to exchange control information using an end-to-end protocol for M&H applications.

6.2.3.6 Application configuration

The terminal is owned by the user. The SSU is operated by the service provider and responsible for connecting the terminal to the selected host. The host and its associated M&H database are operated by the information provider.

Figure 5 below shows the configuration for point of information and point of sales applications.

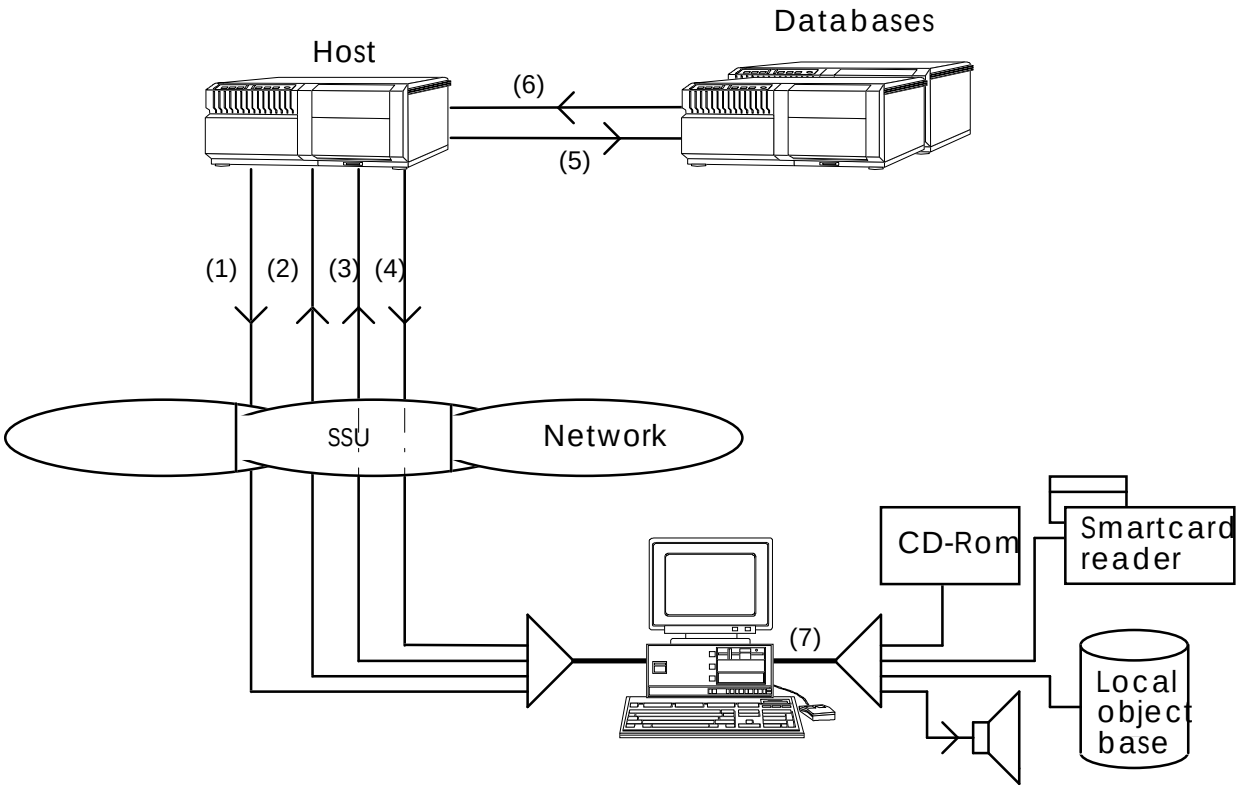

Figure 5: Specific configuration for point of information and point of sales applications

The following communication channels are used in the example figure above:

(1) Application control data from host to terminal;
(2) Application control data from terminal to host;
(3) MHEG objects and data content from terminal to host;
(4) MHEG objects and data content from host to terminal;
(5) MHEG object, or content request from host to database;
(6) MHEG object, or content from database to host;

(7) external input and output devices connected to the terminal.

Application requests for objects that are not present at the terminal result in network requests for the corresponding objects.

In this example the storage of object and application logic may be distributed. The application may start on the terminal, then in a specific application part, the application logic may be taken over by the host. After termination of this application part, the control is returned back to the terminal.

The user's terminals are connected to a network which allows the exchange of commands with a remote application and the exchange of data corresponding to the application. These data can be stored on one or on several databases. From the front-end point of view the user is only working on one database. Data can be stored on the local object base as well. These objects may either by downloaded through the network or distributed by any other means (CD-ROM, floppy disk, etc,...). For the user the storage location of the object is only important with regard to response times and network costs.

The following scenarios can be identified:

- an application uses the same objects in every session (Telebanking, Tourist Guide, etc,...). In this case the application should have a means to store objects in the terminal between sessions and to maintain the objects that belong to it in the terminal (version numbers, date, etc,...);

- the number of objects is very big, e.g. for sales catalogues. In this case the application needs to identify objects that have been supplied previously on a storage media and make them available for the current application.

An important requirement for such applications is to have an interface to the external environment. Input devices such as smartcard readers, VCRs and TV cameras should be accessible. Communication between MHEG objects and such devices should be facilitated by the SIR. An example for the use of such device is a smartcard reader that may be used to give access to a telebanking application.

6.2.3.7 Use of MHEG

The association of attractive audiovisuals, showing the products to best advantage, with the stock control and ordering process is a natural commercial evolution. The use of audiovisual interfaces makes it possible for the public to use computer based real-time information systems. For example a kiosk where information is accessed through Videotex or locally from Compact disk (CD).

MHEG objects contain the hyper-link information as well as all functionality needed to implement a Graphical User Interface (GUI). They contain presentation and navigation information.

6.2.3.8 Use of scripts

In this application example scripts may be used to provide the following functionality:

- arithmetic operations, e.g. price reckonings;

- interface to non-standard external devices, e.g. smart card readers;

- upgrading of the terminal's capabilities, e.g. distribution of new software release, new device drivers, etc,...

6.2.3.9 Application architecture

The application architecture corresponds to the terminal-to-host configuration presented in the general model.

6.2.3.10 API requirements

Table 3 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 3

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	✓
Exception	✓

Table 4 gives an overview of the API functionalities that are required by the host in this particular application.

Table 4

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	✓
Exception	✓

6.2.3.11 SIR requirements

Table 5 gives an overview of the SIR functionalities that are required in this particular application.

Table 5

Function family	Needed
external device control	✓
external device control for data acquisition	
manipulation of MHEG objects	✓
access to external data	
access to external advanced calculation capability	✓
computations, variable handling and control structures	✓

6.2.4 Multimedia retrieval services featuring remote application and remote data

This category comprises all scenarios where the application logic is provided by a host and the multimedia information is provided by a database. Both control and data information is then interchanged through the networks to the terminal.

6.2.4.1 Application example: Interactive telematic training and education services

The following scenarios characterise interactive telematic training and education applications from a user's perspective. A user:

- consults pre-edited multimedia courseware;

- interacts with simulations;

- asks for an explanation of some demonstrated process and gets it through additional hyperlinks in the lessons;

- makes personal annotations about lessons;

- forwards remarks about some aspects of lessons to the lesson preparer or author.

The following application parameters are required:

- user identification;
- access/control priority right management;
- all the user terminals are not on the same network;
- use of several services/media simultaneously;
- variable allocation of bandwidth for the media;
- charging.

6.2.4.2 Service attributes

The following service attributes characterise this category of applications:

- terminal-to-host configuration;
- bi-directional asymmetric communication;
- establishment of the communication around the clock;
- establishment of the communication on demand;
- information is supplied on demand;
- point-to-point connection.

6.2.4.3 Networks

According to the complexity of data, all networks mentioned in subclause 6.2.1.1 are possible candidates except the PSTN which might not offer sufficient data throughput to cope with the frequent transmission of M&H objects. It is most likely that ISDN will be used.

6.2.4.4 Terminal Equipment

Even though a large number of users of this particular application will belong to the residential user group, the TE is likely to be PC-based.

6.2.4.5 Information interchange scenario

MHEG objects are stored on M&H databases connected to the host. To start the lesson the user establishes connection to the SSU and selects the desired application. According to the application part (chapter), specific objects are transmitted to the terminal. There is also a need to exchange control information using an end-to-end protocol for M&H applications. User inputs are sent to the host following specific trigger events.

6.2.4.6 Application configuration

The terminal is owned by the user (student). The SSU is operated by the service provider and responsible for connecting the terminal to the selected host. The host and the associated M&H database are operated by the information (lesson) provider.

Figure 6 shows the application architecture of interactive training and education scenarios.

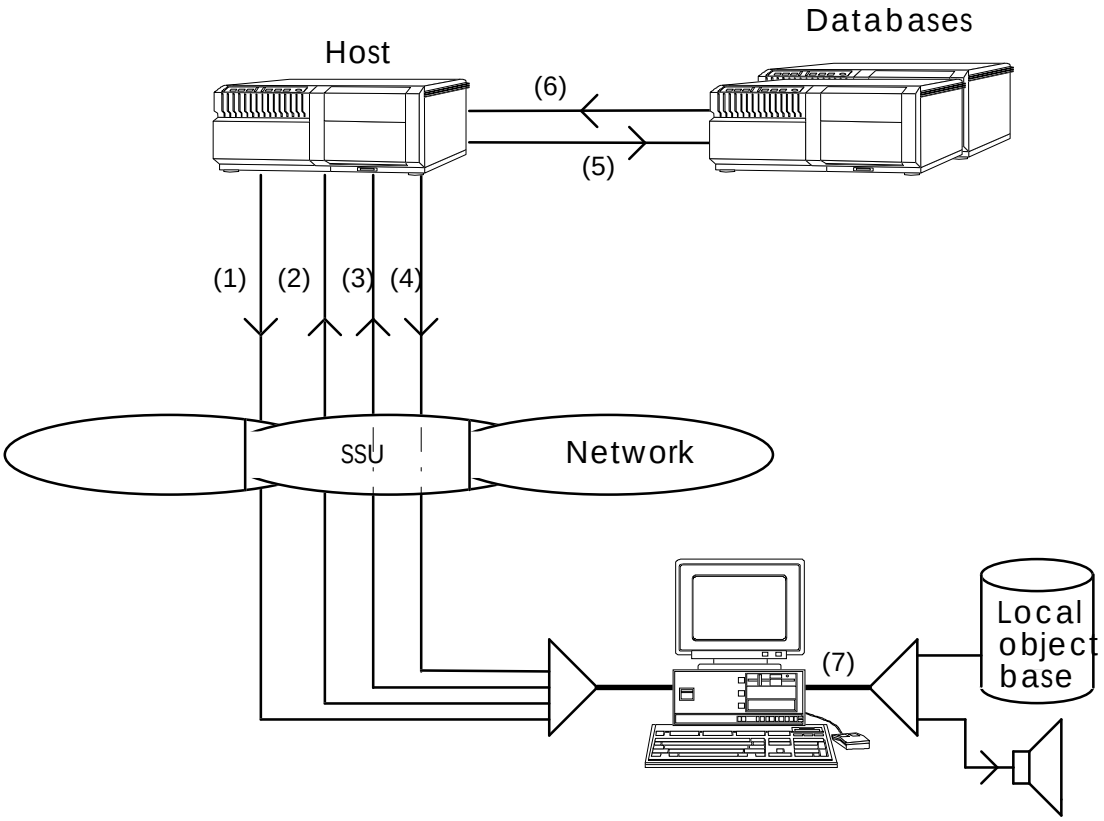

Figure 6: Specific configuration for interactive training and education scenarios

The following communication channels are used in the example figure above:

(1) Application control data from host to terminal;
(2) Application control data from terminal to host;
(3) MHEG objects and data content from terminal to host;
(4) MHEG objects and data content from host to terminal;
(5) MHEG object, or content request from host to database;
(6) MHEG object, or content from database to host;
(7) external input and output devices connected to the terminal.

The terminal provided to a user is connected to a network which allows the exchange of commands with a remote application and the reception of data corresponding to the application. These data can be stored on one or on several databases. From the user's point of view, he is only working on one database. The user can make personal annotations that are stored between sessions on his local database. The MHEG objects are transmitted in real-time. By answering questions and asking for additional information the user can browse his way through the database. Data may be available in different formats and represent different types of media. The local storage of object in this example is not a main feature of the system.

6.2.4.7 Use of MHEG

The extension of data processing and telematics to support audiovisuals makes it a more attractive tool to educators since it improves the interface with the students. There is also a need to exchange audiovisuals between tools and to re-use audiovisuals in other applications. The audiovisuals should be structured in such a way that they can be updated, modified and personalised easily.

MHEG objects contain the hyperlink information as well as all functionality needed to implement a GUI. They contain presentation and navigation information.

6.2.4.8 Use of scripts

In this application example scripts may be used to provide the following functionality:

- interface to non-standard external devices for simulations.

6.2.4.9 Application architecture

The application architecture corresponds to the terminal-to-host configuration presented in the general model.

6.2.4.10 API requirements

Table 6 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 6

Function family	Needed
Session	✓
Directory	
Interchange	✓
Accessor	
Modifier	
Handling	
Behaviour	✓
Exception	✓

Table 7 gives an overview of the API functionalities that are required by the host in this particular application.

Table 7

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	
Exception	✓

6.2.4.11 SIR requirements

Table 8 gives an overview of the SIR functionalities that are required in this particular application.

Table 8

Function family	Needed
external device control	✓
external device control for data acquisition	
manipulation of MHEG objects	
access to external data	
access to external advanced calculation capability	
computations, variable handling and control structures	✓

6.3 Applications of distribution services and VoD

This subclause describes the characteristics of MHEG-using applications which are based on distribution or on-line video retrieval services. It evaluates and analyses their requirements (functional and technical) in terms of MHEG API and script representation.

6.3.1 Common characteristics and taxonomy

This family gathers a range of applications provided to the general public at their home premises. Such applications may cover the whole spectrum of general interest themes (sales and advertising, training and education, entertainment, news and information...). ITU-T Draft Recommendation F.MDS [14] gives examples of applications and scenarios.

The main common feature within the family is therefore its market target - residential users. This leads to several consequences:

a) moving video is a key attraction factor for this market. Moving video should feature at least VHS-quality;

b) interactivity is the major added value of the new services. Interactivity should be simple and intuitive (i.e. easy to learn and remember) rather than rich and complex;

c) the services should be accessed from mass-market, low-cost terminals. Following a) and b), TE should be based mainly on TV and set-top box, interactivity being provided by a remote control pad. However, PC-based platforms should also be envisaged;

d) the services should be accessible via a public network widely available in residential areas, thus based on a physical network with which the general public is connected for either TV distribution or telephone communication;

e) interworking with the currently existing services aimed at the general public, i.e. TV distribution and Videotex-like retrieval services, should be taken into account especially concerning the design of appropriate customer premises equipment.

6.3.1.1 Networks

Different service architectures are possible according to the underlying service and availability at home premises. There are however some common trends:

- the distribution network (terminal access network) is likely to be based (at a first stage) on existing physical networks, either telephone Unshielded Twisted Pair (UTP) or CATV coax cable. In the former case (interactive network basis), distribution capabilities may be provided using ADSL or Hybrid Digital Subscriber Line (HDSL). In the latter case (distribution network basis), there are different possible architectures, some of them including fibre, interactive capabilities being provided using either PSTN modem or cable return channel. Fibre To The Home (FTTH) Passive Optical Network (PON) and wireless access networks may be used at a later stage;

- the transport network (database access network) is likely to be based on Asynchronous Transfer Mode (ATM) whenever high and flexible bandwidth is required, i.e. for multimedia information retrieval services.

6.3.1.2 Terminal Equipment

The TE should integrate itself as easily as possible in the existing home environment and should, therefore, be based on TV set, set-top-box and remote control pad, together with the appropriate network interfaces. According to the application, this basis may either be enhanced by additional equipment such as a smart card reader, VCR, digital mass storage device, or replaced with microcomputer-based equipment.

6.3.1.3 Service attributes

The family of applications can be divided in three categories, according to the attributes of the underlying services (ITU-T Recommendation I.211 [15]):

- distribution with local interactivity: these applications rely on new versions of the plain old TV distribution service featuring MPEG-2 digital video encoding and multiplexed streams of multimedia data;

- distribution combined with interactivity: in these applications, use of the former service is completed by use of a retrieval service, especially for transaction purposes;

- on-line retrieval: these applications use a service dedicated to the on-line distribution of audiovisual information on a broadband network.

6.3.2 Multimedia distribution services featuring local interactivity

This category brings together applications of the regular TV distribution service which are made possible by the digital encoding of broadcast audiovisual information. Multimedia information is being multiplexed with the audiovisual information and transferred to the terminal. The user can then navigate through this information.

The selected pilot application in this category is the Electronic Programme Guide (EPG) or TV guide.

Among other application examples in this category, the "broadcast multimedia magazine" also receives significant interest from the actors of the home market.

6.3.2.1 Application example: EPG

The EPG application is a basic one which is made necessary by the availability of many channels that the use of digital compression techniques in video broadcasting will soon make possible. This application both aims at helping users to find their way across the plethora of available events and at helping broadcasters to promote their programme supply and their public image.

The user of this application can ask for the presentation of information about the programmes and events on his TV screen. He can navigate through this information using his remote control pad. This information may be structured and presented to the user in several ways, according to the application design. In addition to traditional TV functions (audiovisual display, zapping, access control), the user can for instance:

- select a channel from a mosaic of available programmes within the whole network or within a bouquet;

- consult the timetable of next events within one or several programmes;

- ask for the display of information (duration, theme, target public, summary, language...) on a given event;

- search among the next events according to one or several criteria (theme, type of event, target public, language...), e.g. using scrolled lists;

- program his VCR so as to record a selected event;

- etc,...

6.3.2.2 Service attributes

The service attributes are the following:

- distribution with individual user presentation control;
- user-to-host;
- on demand (selection);
- unidirectional;
- access around the clock;
- broadcast/multicast.

6.3.2.3 Networks

The distribution network should be any network that is used for TV distribution and allows the transmission of digital information.

6.3.2.4 Terminal Equipment

The basic TE is based on regular TV and specific set-top-box.

6.3.2.5 Information interchange scenario

The interactive information may be transferred either in a specific channel or in a data stream multiplexed within a channel, using several modes:

- it may be broadcast in a cyclic mode. This is the ideal mode as long as the channel bandwidth allocated to the information makes it possible to satisfy the combined requirements regarding the information size, access frequency and access delay. In this mode, the terminal can nevertheless perform local storage either in a lasting (i.e. up to the next version of the object) or temporary (i.e. up to the end of the session) fashion to improve further access to this particular information;

- it may be downloaded if needed, either at a pre-set time or according to a downstream signalling protocol.

6.3.2.6 Service configuration

The different roles of the application are:

- programme providers (broadcasters): they both act as application and information providers; they are called service providers in the Digital Video Broadcasting (DVB) vocabulary;

- the network operator (carrier), which may also act as an application provider e.g. if there is a reserved EPG channel; with respect to the ITU-T vocabulary, the network operator is the service provider;

- users.

The service configuration involves a host and a terminal.

The presentation objects are provided by the programme provider.

The host is provided by the programme provider.

The terminal is provided by the user, although he will usually buy or rent it to either the service operator or an independent vendor, in the latter case according to specifications established by the service operator.

6.3.2.7 Use of MHEG

MHEG provides an appropriate way to represent the broadcast presentation and navigation information. It complements the standards for the transfer and encoding of audiovisual, access control and event information data by providing a value-added multimedia man-machine interface. MHEG meets the requirements since interchange takes place in real-time (though presentation does not necessarily) and terminals cannot cope with non-final form objects due to their minimal (processing or storage) resources.

MHEG objects are interchanged between host and terminal via the distribution network.

MHEG objects express presentation and navigation of the EPG information. They can also express part of the information (e.g. event summary) as well as some additional presentation (e.g. programme provider logo). They should be pre-edited, but some automatic updating may take place.

MHEG objects may be stored on the terminal mass storage, whether pre-loaded (if MHEG is used as the representation means for the terminal's built-in interface) or stored in a lasting or temporary fashion.

6.3.2.8 Use of scripts

The application may need to interchange scripts to provide several added-value functions:

- to allow external device control, e.g. VCR programming (the user navigates through the TV Guide and selects an event for recording). It is not yet clear whether a specific SIR is mandatory or MHEG-1 specific profiles could be actually used for controlling external devices (other than regular presentation devices);

- to perform some computations such as interpretation of database-like requests. The MHEG representation is not appropriate for the data on which such computations apply. Scripts should then rather use the application-specific data being transferred within the multiplex in non-MHEG form, though MHEG objects would still provide the display of, and navigation through, this information;

- (unlikely in this application context) to upgrade the terminal's capabilities (e.g. distribute new software release, new device drivers...).

6.3.2.9 Application architecture

The functional architecture which applies to this application is the one appropriate for terminal-to-host configurations (see figure 1).

6.3.2.10 API requirements

Table 9 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 9

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	
Modifier	✓
Handling	
Behaviour	✓
Exception	✓

Table 10 gives an overview of the API functionalities that are required by the host in this particular application.

Table 10

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	
Exception	✓

6.3.2.11 SIR requirements

Table 11 gives an overview of the SIR functionalities that are required in this particular application.

Table 11

Function family	Needed
external device control	✓
external device control for data acquisition	✓
manipulation of MHEG objects	✓
access to external data	✓
access to external advanced calculation capability	
computations, variable handling and control structures	✓

6.3.3 Multimedia distribution services featuring real-time terminal-to-host interactivity

These applications extend the category described above by adding remote interaction capabilities to the user terminal. During or after having navigated through the broadcast information, the user can interact with the host, e.g. to achieve a transaction.

The selected application in this category is teleshopping, also called home shopping.

Another application pertaining to this category is staggered Video on Demand (sVoD).

6.3.3.1 Application example: teleshopping

When the user selects a particular teleshopping programme channel using his regular TV equipment, he can access the following functions:

- view audiovisual broadcast advertising consisting for instance of commercial presentations;

- navigate through a hypermedia catalogue (including especially extensive information on the currently advertised product);

- fill in and forward product order forms.

Among possible providers:

- mail order companies (selling general purpose home equipment, e.g. refrigerator, shoes);
- supermarkets;
- travel agencies (e.g. hotel rooms, flights);
- real estate agencies.

6.3.3.2 Service attributes

The service attributes are the following:

a) for the distribution service:
   1) distribution with individual user presentation control;
   2) user-to-host;
   3) on demand (selection)/permanent;
   4) unidirectional;
   5) access around the clock;
   6) broadcast/multicast.

b) for the interactive service:
   1) retrieval;
   2) user-to-host;
   3) on demand;
   4) bi-directional asymmetric;
   5) user identification required;
   6) partial encryption required;
   7) point-to-point.

This service attributes description assumes that two separate services are being used and accessed from the TE. However, depending on the operator's involvement, there might be only one (combined) service. Data broadcasting and real-time audiovisual broadcasting may also be provided as two separate services by the operator.

6.3.3.3 Networks

The distribution network may be any network used for TV distribution and allowing the transmission of digital information. The interactive access may be provided either via the cable return channel or (more likely) via the PSTN.

6.3.3.4 Terminal Equipment

The basic TE is based on regular TV and set-top-box. This basis should be completed by integrating a modem (for interactive network access), some authentication device (e.g. smart card reader) allowing to control transactions, and possibly some local mass storage device, either rewritable (e.g. digital tape, flash memory extensions) or not (e.g. CD-ROM).

6.3.3.5 Information interchange scenario

This application mixes broadcast advertising and data downloading. While the audiovisual sequence is broadcast, presentation objects multiplexed on the same channel are being interchanged between host and terminal via the distribution network, possibly also between terminal and host via the interactive access network.

Presentation objects may be interchanged in three ways:

- non-real-time interchange (i.e. downloading) of the catalogue structure (the "hypermedia web") and other key elements such as the order form. Hypermedia links may be accessed at any time from everywhere. Cyclic access delay should be limited to one search (i.e. getting the right multimedia page). This information may be updated e.g. once a day;

- on-line cyclic broadcasting of the multimedia entries matching the catalogue "pages". The pages corresponding to (or somehow linked to) the product currently been advertised by the broadcast commercial presentation have higher priority, which means there that they may be broadcast several times more often than others;

- the completed order form may be forwarded in MHEG format so as to allow further consultation. However, such MHEG objects should base on an appropriate non-media data format to allow subsequent handling.

Some MHEG objects will be stored on the terminal mass storage in a lasting or temporary fashion.

6.3.3.6 Service configuration

The actors are:

- programme providers (broadcaster): the home shopping channel provider, is both the application and information provider (service provider in DVB terms);

- the network operator (carrier);

- users.

The service configuration involves a terminal, a host and a database.

The host and the database are provided by the programme provider.

The terminal is provided by the user.

The MHEG objects are provided by the programme provider.

The information contents are stored on a database which is provided by the programme provider. Unlike the first category of applications, the database is likely to manage not only the catalogue information but also the pre-recorded audiovisual sequences that are being broadcast (live broadcast on teleshopping channel being quite unlikely).

The MHEG objects used by the host application may be either stored on the host or on the database.

6.3.3.7 Use of MHEG

MHEG provides an appropriate way to represent the broadcast presentation and navigation information. It completes the standards for the transfer and encoding of audiovisual, access control and event information data by providing a value-added multimedia man-machine interface. MHEG is adequate for the need since interchange takes place in real-time (though presentation does not necessarily) and terminals cannot cope with non-final form objects due to their minimal (processing or storage) resources.

MHEG objects are interchanged between host and terminal via the distribution network.

MHEG objects are pre-edited. They express presentation and navigation of the catalogue. How the catalogue information (prices, product codes...) itself is represented is an open issue.

6.3.3.8 Use of scripts

The application may need to interchange scripts:

- to allow external device control, e.g. of a smart card reader (the user navigates through the catalogue and validates his order form using a pre-defined code or credit card transaction);

- to perform some computations such as interpretation of database-like requests, possibly with some intelligent natural or visual language interpreter (e.g. show me all refrigerators manufactured by European companies and priced at less than 500 ECU), order form reckoning (e.g. how much do I have to pay, shipping and taxes included). To allow these computations, the non-media data representation of the catalogue (other than links and presentation) information should be appropriate.

6.3.3.9 Application architecture

The application architecture corresponds to the terminal-to-host configuration presented in the general model.

Transaction management (which might be considered as some sort of script, although this is not relevant to SIR purposes) consists of two functions:

- real-time acknowledgement of the transaction;
- provision for subsequent (manual or automatic) dealing with the user order (charging, sending...).

The terminal is functionally similar to the base terminal for the EPG application, but incorporates some additional features, for instance:

- it integrates an external device interface (to smart card reader);
- the access agent also provides interactive terminal-to-host exchange for transactions;
- the access agent may provide access to additional external mass storage (e.g. Digital Audio Tape (DAT), CD-ROM).

6.3.3.10 API requirements

Table 12 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 12

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	✓
Exception	✓

Table 13 gives an overview of the API functionalities that are required by the host in this particular application.

Table 13

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	
Exception	✓

6.3.3.11 SIR requirements

Table 14 gives an overview of the SIR functionalities that are required in this particular application.

Table 14

Function family	Needed
external device control	
external device control for data acquisition	✓
manipulation of MHEG objects	✓
access to external data	✓
access to external advanced calculation capability	
computations, variable handling and control structures	✓

6.3.4 Multimedia services featuring on-line video retrieval

On-line video retrieval services are a specific instance of M&HIRS (see ETR 084 [12]), also sometimes referred to (improperly) as "broadband Videotex".

This is a new generation of retrieval applications aimed at the home market, especially resulting from the combined technological progress in digital video compression, high-speed interactive networks and microcomputer hardware. They will usually provide their users with:

- information (encyclopaedia, news);
- shopping, reservation services;
- training;
- entertainment (video, games).

As for Videotex, some of these applications will also address the business market, whose requirements (access networks, TE) should therefore also be taken into account.

The most representative M&HIRS application is interactive Video on Demand (iVoD).

Other significant applications aimed at the residential market and using on-line video retrieval services include:

- tele-karaoke;

- multimedia catalogue consultation (with transactional functions such as reservation or purchase);
- news on demand;

- see also ETR 228 [16] and ETR 176 [17].

6.3.4.1 Application example: iVoD

A VoD application provides residential users with the ability to select among a catalogue of pre-recorded programmes (films, news bulletins, sport events, music clips, documentaries, previews), to receive the chosen programme on a TV set and navigate through it using control commands.

The catalogue of available programmes may have a tree or multi-criterion structure. It may federate the catalogues of several content providers (i.e. programme providers). The service operator may also act as one content provider.

To access the application, the user needs to press a channel selection or specific service key on his remote control pad. Navigation through the catalogue is provided via a graphical man-machine interface whose features may depend on the terminal's capabilities. Text, geometric figures and icon display, interaction with scrolling lists and buttons are the most common features. Display of trailers of the programme may or may not be provided by the application.

The service allows to book a programme beforehand so as to reserve the resources for a specified time. During the programme retrieval, navigation through the programme is allowed using functions which resemble VCR commands (play, fast forward, rewind, pause...) possibly with extensions enabled by digital coding (e.g. direct access to a sequence).

This application is either a specific instance or a using application of a more general multimedia information retrieval service (see ETR 228 [16]). However, unlike most existing retrieval applications, VoD is especially oriented towards residential users and requires specific elements in its architecture to cope with the bandwidth requirements of moving video. M&HIRS TE should anyway provide access to other retrieval services (e.g. a "traditional" Videotex service) of interest to residential users.

6.3.4.2 Service attributes

Connection to the service is established on demand. The service should be accessible around the clock. It is user-to-host, point-to-point and bi-directional asymmetric. The iVoD underlying service therefore clearly belongs to the retrieval services family (see ETR 181 [10]).

User identification and authentication are required. This may lead to some information encryption. The service should provide facilities for charging. Protection against copy should also be provided to preserve the content providers' interests.

Interworking with other retrieval services and transactional capabilities are required.

6.3.4.3 Networks

Candidate networks for such a service include the following:

- terminal access may be provided by the PSTN (e.g. through a Videotex Access Point (VAP) with Packet Switched Public Data Network (PSPDN) as transport network), ADSL, HDSL or CATV return channel;

- host-to-terminal sporadic transfer of MHEG objects during the transactional phase may be provided by ADSL, HDSL or CATV in digital mode;

- host-to-terminal Constant Bit Rate (CBR) transfer of audiovisual sequences during the retrieval phase (content distribution) may be provided by ADSL, HDSL, satellite or CATV in digital mode;

- host access and host-to-database communication may use PSDN, Narrowband-ISDN (N-ISDN) or the transport network;

- database access (content transport) may use B-ISDN, Fibre Distributed Data Interface (FDDI) or ATM-based LAN.

6.3.4.4 Terminal Equipment

The typical user terminal consists of a digital TV set combined with an interaction device (usually a remote control pad) and a dedicated set-top-box which is in charge of all logical functions (other than TV display), integrating a communication device for navigation functions, a smart card reader (optional) for identification and authentication purposes, as well as such extensions as VCR, CD-ROM (or CD-I), game console, etc,...

The terminal may also be based on a personal computer equipped with appropriate reception, decoding and communication devices.

6.3.4.5 Information interchange scenario

A session usually consists of two different phases, possibly separated in time:

- the transactional phase is a database navigation application. The results of user interaction are transmitted from terminal to host, possibly as MHEG objects. Multimedia presentation information is then transmitted from host to terminal. It consists of MHEG objects expressing text, graphics and still, also possibly audiovisual sequence, display as well as requests for selection. Information flows are sporadic, so this phase should use variable bitrate channels to allow resource optimisation. The outcome of the phase is the transaction which specifies that the selected programme will be sent to the user's terminal at the specified time;

- the retrieval phase consists in transmitting the requested programme to the terminal. This uses a constant bitrate appropriate to the perceptive quality of the audiovisual sequence. Casually, user requests for navigation inside the programme will be transmitted from terminal to host, possibly as MHEG objects (or also possibly MPEG Digital Storage Media - Control Commands (DSM-CC). This will lead to some processing and influence the audiovisual sequence being transferred to the terminal.

6.3.4.6 Application configuration

The actors are:

- information provider: makes the titles available;
- application provider: provides the application;
- service operator: provides access to the service;
- user.

The front-end (SSU) is provided by the service operator. The host (application server) is provided by the application provider. The database (contents server) is provided by the information provider. The terminal is provided by the user.

MHEG objects expressing navigation through the application (e.g. directory) are provided by the application provider.

NOTE: In other M&HIRS applications, MHEG objects expressing the information itself are provided by the information provider.

The information contents are stored on video databases.

The directory of available titles and the information access procedures are supplied to users through the host.

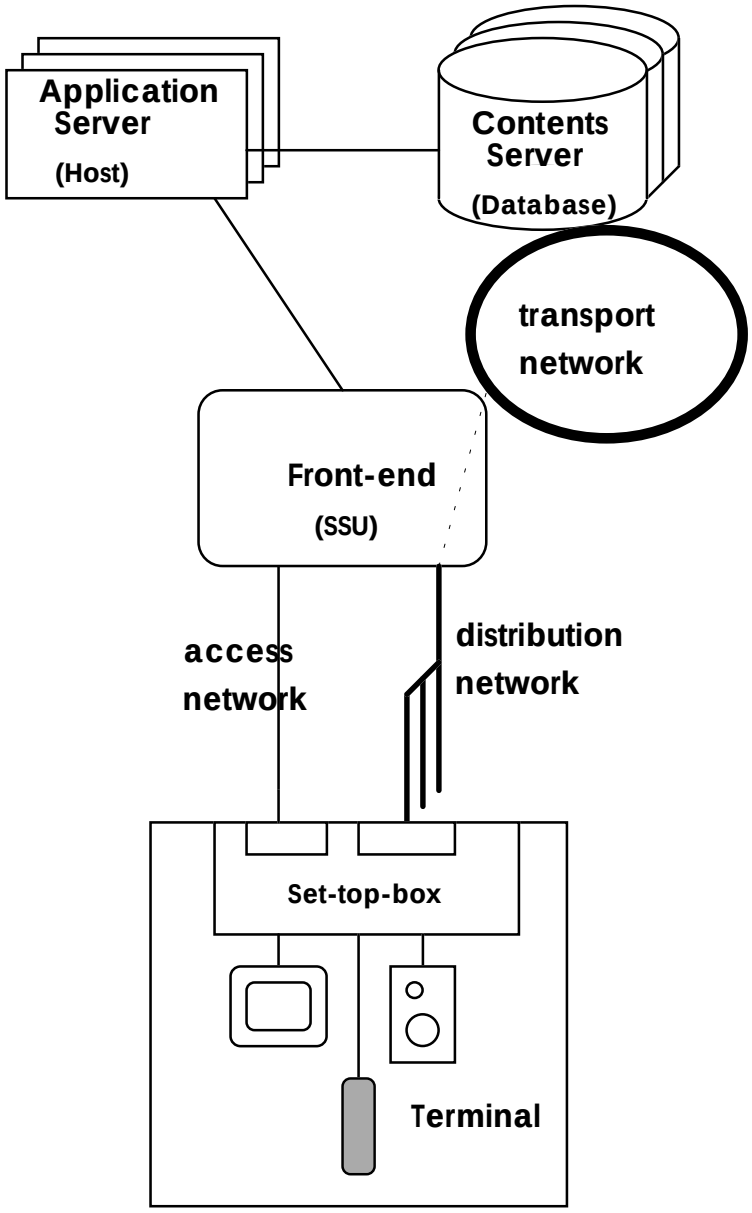

Figure 7: Application configuration

The service front-end receives incoming calls from the user. The communication is then established between terminal and front-end. As the SSU, the front-end performs the access control and charging functions.

The front-end then opens an interactive dialogue channel between the terminal and application server (host), with which the terminal communicates for the transaction phase. MHEG objects are transmitted between terminal and host; there should therefore be MHEG engines at these locations. It uses an application script (program), MHEG objects intended for presentation to the user and directory information possibly stored in a database. This directory information needs to be frequently updated. During the transaction phase, trailers may be sent to the terminal. It is then likely that a connection similar to that used in the retrieval phase is set up.

For the retrieval phase, a distribution (unidirectional) channel is opened from the contents server (multimedia database) to the terminal to transmit the video data. A command channel is opened between the application server and the contents server. A transport channel is opened between the contents server and the front-end (acting as a gateway between the transport and distribution networks).

6.3.4.7 Use of MHEG

MHEG objects are interchanged bi-directionally between host and terminal via the interactive access network, possibly also via the distribution network (if different).

MHEG objects express navigation and presentation information:

- during the information selection phase, navigation through the directory of available titles (directory presentation, trailers, user selection, user transactional information);

- during the retrieval phase, presentation and structure of the retrieved information;

- in other M&HIRS applications, navigation through the information may also occur during the information retrieval phase.

6.3.4.8 Use of scripts

It is useful to interchange scripts to allow external device control, e.g. smart card reader (the user validates his choice of a programme using a card transaction) or VCR programming/triggering.

In M&HIRS applications other than VoD, scripts may be interchanged also:

- to compute game results;
- to perform local simulations based on user-provided data;
- etc,...

6.3.4.9 Application architecture

The application architecture corresponds to the terminal-to-host configuration presented in the general model.

Transaction management consists in acknowledging and recording the order and preparing requests (toward the databases and toward the service resource management and charging management units) for dealing with it.

6.3.4.10 API requirements

Table 15 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 15

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	✓
Exception	✓

Table 16 gives an overview of the API functionalities that are required by the host in this particular application.

Table 16

Function family	Needed
Session	✓
Directory	✓
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	
Exception	✓

6.3.4.11 SIR requirements

Table 17 gives an overview of the SIR functionalities that are required in this particular application.

Table 17

Function family	Needed
external device control	✓
external device control for data acquisition	✓
manipulation of MHEG objects	✓
access to external data	✓
access to external advanced calculation capability	
computations, variable handling and control structures	

6.4 Applications of conversational services

This subclause describes the characteristics of MHEG-using applications which are based on conversational services. It evaluates and analyses their functional requirements in terms of MHEG API and script representation.

There are three service types that can be classified as multimedia in this way, and that will be described in greater detail. These three service types are as follows:

- videotelephony;
- videoconferencing;
- real-time control (or telecontrol).

The Videotelephony service provides communication for the transfer of voice/sound, moving pictures, and video scanned still images and documents between two locations (e.g. teleeducation, teleshopping, tele-advertising).

The service attributes are:

- person-to-person;
- demand/reserved/permanent;
- point-to-point/point-to-multipoint;
- bi-directional symmetric/bi-directional asymmetric.

The Videoconference service provides multipoint communication for the transfer of voice/sound, moving pictures, and video scanned still images and documents between two or more locations (e.g. teleeducation, teleshopping, tele-advertising).

The service attributes are:

- person-to-group;
- group-to-group;
- demand/reserved/permanent;
- point-to-point/point-to-multipoint;
- bi-directional symmetric/bi-directional asymmetric.

The real-time control (telecontrol) service provides multipoint communication for the transfer of data, still and moving images; and for real-time control between one site and a number of remote sites.

The service attributes are:

- person-to-machine;
- demand/reserved/permanent;
- point-to-point/point-to-multipoint;
- bi-directional symmetric/bi-directional asymmetric.

6.4.1 Common characteristics and taxonomy

This family brings together a range of applications for domestic users and for businesses alike. Such applications cover areas as videotelephony, videoconferencing, real-time control, outside broadcasting and games networks.

The main common features within the family is real-time user-to-user communication, although some users can be machines (such as ovens) or processors (for games). Other significant aspects of the family, are as follows:

- communication between users is generally symmetrical when the users are human, otherwise it tends to be asymmetrical with a greater bandwidth of information coming from the machine than going to it;

- the communication configuration can be a point-to-point (videotelephony), consecutive point-to-multipoint (videoconferencing) or point-to-multipoint (real-time control) and there can be a mixture of configurations in both directions of transmission (i.e. in real-time control, the control direction is consecutive point-to-point, whereas the monitor direction is multipoint-to-point);

- for most of the TE used in the applications (other than for videoconferencing equipment or real-time control equipment, etc,...), the equipment used is a multifunctional PC platform;

- the applications should be accessible via a public network available in residential areas and business communities.

6.4.1.1 Networks

The speed and availability of each of the applications is different depending on the availability of networks. The types of networks that can be used, is given in table 18:

Table 18

Multimedia Conversational Service	Target User Group	Possible Networks Used
Videotelephony	Domestic and business	PSTN, ISDN, B-ISDN
Videoconference	Business	ISDN, B-ISDN
Real-time control (telecontrol)	Domestic and business	PSTN, ISDN, B-ISDN

ISDN is the ideal network minimum requirement for these applications, although PSTN can be used with a modem (V.34 gives 28,8 kbit/s full-duplex over a switched connection) but this will restrict the application. Switched Multimegabit Data Service (SMDS), Connectionless Broadband Data Service (CBDS) and Frame Relay networks are not suitable as they are not isochronous by nature and cannot easily communicate voice and video.

6.4.1.2 Terminal Equipment

For domestic use, the TE should be either a low-cost videotelephone or a multifunctional PC. For business use, other than for special videoconference equipment etc,... a PC platform should be used.

For Videotelephony dedicated videotelephones or PCs with videotelephony capabilities at either end of connection will be used. For Videoconferencing dedicated videoconference equipment or (exceptionally) PCs with videoconference capabilities will be used. In non point-to-point sessions, an MCU will be used (as part of the PNO videoconference service). Real-time control (telecontrol) services will use dedicated terminals or PCs at human end of connection with dedicated telecontrol unit at either end. Connected to the telecontrol unit will be sensors, cameras, transducers and configurable devices, etc,...

6.4.1.3 Service attributes

Conversational services can be categorised into four distinct categories, as follows:

1) moving pictures (video) and sound;
2) sound;
3) data;
4) document.

Of these only 1) contains services that can be classified as multimedia, although in certain applications all four categories of service can be included in a single session.

The three applications can be categorised according to table 19:

Table 19

Application	From controlling users point of view	From other users point of view
Videotelephony	conversational service	conversational service
Videoconference	conversational service with local interactivity and remote retrieval	conversational service
Real-time control (telecontrol)	conversational service with local interactivity and remote retrieval	conversational service

6.4.2 Multimedia conversational services featuring videoconferencing

For videoconferencing service, this subclause gives an explanation of the application and describes the service attributes, applicable networks and TE, the service and functional architecture and the requirements for, and use of, MHEG and scripts.

The service can be viewed as an extension of the Audiovisual Conferencing Service (AVCS) using MHEG multimedia objects and real-time video.

6.4.2.1 Application example: multimedia videoconferencing

The following features characterise applications based on the videoconference service from a user's perspective. A participant of a videoconference can:

- communicate with the other participants seeing their talking head image on his terminal device;
- establish private connections with other participants;
- perform joint document editing together with other participants;
- interchange documents or data with other participants;
- perform database enquiries to a connected database;
- get system information on the ongoing conference.

A special role in a videoconference is assigned to the conductor. The conductor, if present in a conference, is a participant who controls certain aspects of the conference: control of application invocation, control of communication between conductor-controlled applications, control over conference participants and conference termination. There is either zero or one conductors in a conference. A

participant becomes conductor by grabbing the conductor token after connecting to the conference, or by requesting or accepting conductorship from the current conductor.

The main service attributes/parameters are as follows:

- each PC or workstation user can see and/or modify the same window content, whether it be graphics, text, pictures, moving pictures or any combination of them;

- all content data is not necessarily shared by all conference locations, as for example, some multimedia conference systems would not have videoconferencing, but only multipoint telephony and other data facilities;

- the conference room can be equipped to display the media types (i.e. text, graphics, images, etc,...) shared by the PC/workstations.

6.4.2.2 Service attributes

The service attributes are the following:

- person-to-person, person-to-group, group-to-group;
- demand/reserved/permanent;
- point-to-point/point-to-multipoint;
- bi-directional symmetric/bi-directional asymmetric;
- interworking with information and retrieval and messaging services (including file transfer).

6.4.2.3 Networks

Ideally ISDN (typically three basic access of N-ISDN, equivalent to 384 kbit/s of digital bandwidth) or B-ISDN, should be used for this application. The use of LANs as terminal access networks is likely. The use of PSTN is unlikely as its performance would be frustrating for the users.

6.4.2.4 Terminal equipment

This application is primarily for business use. Either dedicated videoconference equipment or workstations (including PC-based platforms) should be used.

6.4.2.5 Information interchange scenario

The Multipoint Conference Unit (MCU) takes the incoming data stream from the user that has the baton and distributes the data stream to all users. The conductor notifies the MCU as to which user has the baton so it can replicate the appropriate data stream.

The database stores all forms of information (including the MHEG objects containing the M&H information) which can be retrieved or updated as part of the videoconference session (via the MCU). MHEG objects can be generated at the terminal in real-time as well and distributed via the MCU.

6.4.2.6 Application configuration

The AVCS description defines the roles of the actors, but in general:

- all users can act as application and information providers with the conductor having a special role;

- some users (especially those in videoconference studios) can only act in a restricted manner and are seen and heard but cannot manipulate data, text graphics or other media.

The presentation objects are provided by all users (on a basis decided by the conductor).

The MCU is provided by the network operator.

The TE is provided by the user, although the equipment itself may be bought, rented or leased. Videoconference studio equipment can also be bought, rented or leased, although very often the studio is

hired for a small duration especially for the conference. The studio could belong to the network operator or some other organization.

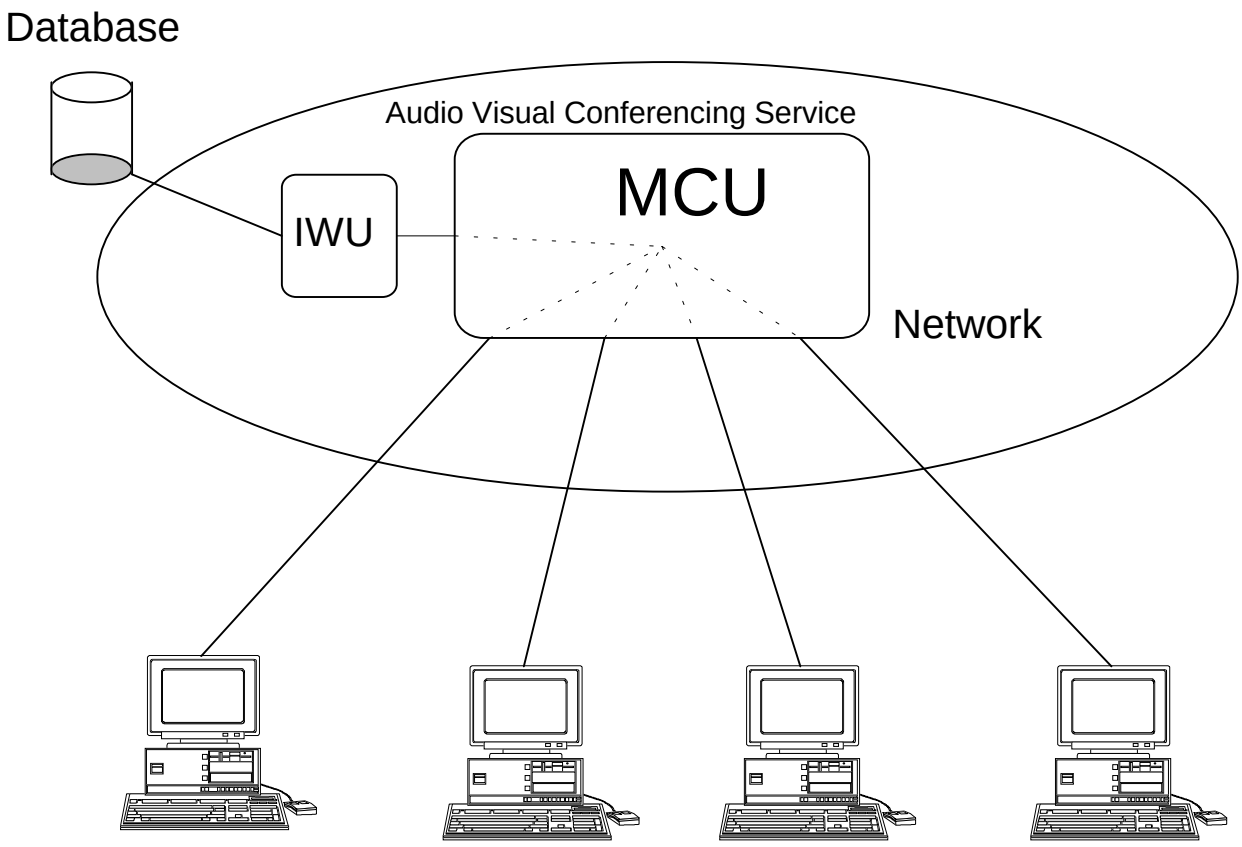

Figure 8: Application configuration

This multimedia conferencing service is based on PC or workstations in a multipoint configuration. A MCU is essential for multipoint configurations. The information is transmitted over ISDN (typically three basic access of N-ISDN, equivalent to 384 kbit/s of digital bandwidth) and will use ITU-T Recommendation T.122 [18] to orchestrate the session. Interworking capabilities with a videoconferencing room can also be included or could be part of a videoconference session. Each participant in a multimedia conference can be connected to a public database. Local storage (CD-ROM, CD-I, etc,...) can also be used.

6.4.2.7 Use of MHEG

MHEG objects are interchanged between users via the MCU or to/from the database which stores MHEG objects. The MHEG objects stored on databases are most likely to be pre-edited whereas the objects interchanged between the terminals may be created in real-time. During a conference objects may be updated (e.g. agenda).

6.4.2.8 Use of scripts

The use of scripts in videoconference is regarded as being rather unlikely.

6.4.2.9 Application architecture

The application architecture corresponds to the terminal-to-terminal configuration presented in the general model.

6.4.2.10 API requirements

Table 20 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 20

Function family	Needed
Session	✓
Directory	
Interchange	✓
Accessor	✓
Modifier	✓
Handling	
Behaviour	✓
Exception	✓

6.4.3 Multimedia conversational services featuring videotelephony

For the videotelephony service, this subclause gives an explanation of the application and describes the service attributes, applicable networks and TE, the service and functional architecture and the requirements for, and use of, MHEG and scripts.

6.4.3.1 Application example: Interactive games based on the videotelephony service

The following features characterise interactive applications based on the videotelephony service from a users perspective. A user can:

- communicate with the other user seeing his talking head image on his terminal device;
- play an interactive game involving real-time interactions of both users (e.g.: tennis simulation).

Another videotelephony application is tele-tutoring: a user (tutor) can take over the other user's screen display, exchange files, talk with the other user, see his talking head, etc,...

6.4.3.2 Service attributes

The service attributes are the following:

- person-to-person;
- demand/reserved;
- point-to-point;
- bi-directional;
- symmetric.

6.4.3.3 Networks

Ideally ISDN (typically one basic access of N-ISDN, equivalent to 128 kbit/s of digital bandwidth) or B-ISDN should be used for this application. If two B-channels are used, the video image would occupy one B-channel and the voice and other information, the second B-channel.

The use of PSTN is possible although the performance using currently available encoding techniques would be frustrating for the users.

6.4.3.4 Terminal Equipment

This application is for an domestic use. Either dedicated videotelephony equipment or PC-based platforms should be used.

6.4.3.5 Information interchange scenario

In addition to the services offered by videotelephony, MHEG objects enabling the simulation game need to be interchanged between the two terminals. The application logic for the game is based on one terminal. After the establishment of the connection the users might wish to start the game. Then the objects that represent the game environment and the players need to be transmitted to the other terminal. Both of the users may then interact on the objects that represent the tennis players. The interaction needs to be exchanged in real-time to make sure that both of the users have the same situation on their screens. The MHEG objects may contain sound data to make the simulation more realistic, as well as graphical data to represent the players and the game environment.

6.4.3.6 Application configuration

The different roles of the application are as follows:

- both users can act as application and information providers.

The TE is provided by the user, although the equipment itself may be bought, rented or leased.

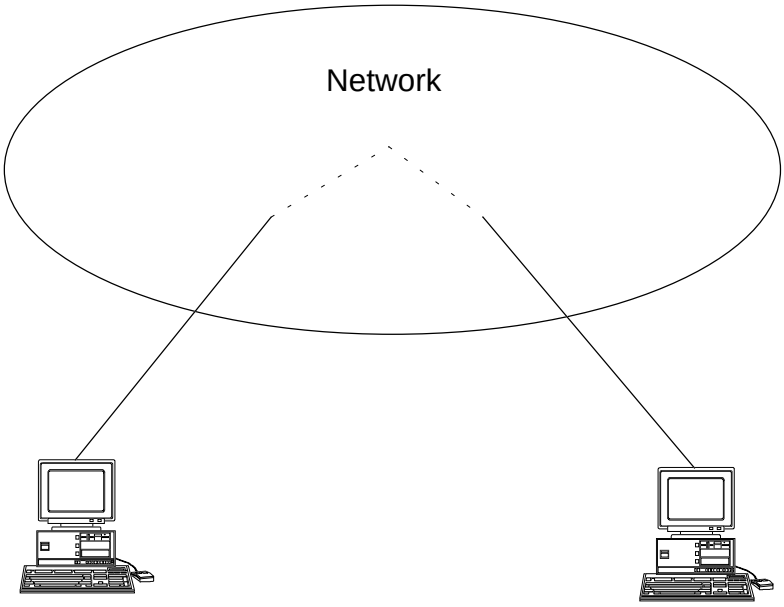

Figure 9: Application configuration

Videotelephony is generally point-to-point between two users only. If ISDN supplementary services are used Three-Party supplementary service (3PTY) or Conferencing supplementary service (CONF), the application configuration would be similar to AVCS.

6.4.3.7 Use of MHEG

MHEG provides the appropriate way to implement and represent the objects needed for the simulation. During a game the objects are interchanged and manipulated.

MHEG objects express the game environment as well as the players. The basis for the game is the user interaction on the objects. The actions applied to the objects to change them according to the game situation are interchanged as objects as well.

6.4.3.8 Use of scripts

Scripts are used to control the players. An rt-script is associated with each rt-object to control its movement and presentation. For that purpose, advanced calculation facilities and access to the MHEG API primitives are needed.

6.4.3.9 Application architecture

The application architecture corresponds to the terminal-to-terminal configuration presented in the general model.

6.4.3.10 API requirements

Table 21 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 21

Function family	Needed
Session	✓
Directory	
Interchange	✓
Accessor	✓
Modifier	✓
Handling	✓
Behaviour	✓
Exception	✓

6.4.3.11 SIR requirements

Table 22 gives an overview of the SIR functionalities that are required in this particular application.

Table 22

Function family	Needed
external device control	✓
external device control for data acquisition	
manipulation of MHEG objects	✓
access to external data	
access to external advanced calculation capability	✓
computations, variable handling and control structures	✓

6.4.4 Multimedia conversational services featuring user-to-machine communication

For real-time control service, this subclause gives a full explanation of the application and describes the service attributes, applicable networks and TE, the service and functional architecture and the requirements for, and use of, MHEG and scripts.

6.4.4.1 Application description: real-time control for domestic or business premises

In this, application control and monitoring of domestic and/or business premises is made possible. Connection to successive sites allows a single human controller to check and regulate temperature and air conditioning in each site; or to open and close doors and windows; or even to monitor road traffic and indicate diversions. Video surveillance may be used as part of this application.

The human controller can continuously monitor each of the sites and only change any settings at a site when necessary. For this reason the type of network connection for each site can be split into the monitoring function (telesurveillance) and the control function (tele-action).

Attached to each site controller is storage, which holds the characteristics of the site and other information related to the site. In this way, each site can be accessed by a number of different human controllers without the need for each terminal to store every single detail of all possible sites.

6.4.4.2 Service attributes

In this application, the monitoring and control activities, for each site, may be configured differently, as shown in table 23:

Table 23

Network Configuration for monitor and control functions	Monitoring Activity	Control Activity
As Needed	Point-to-Point Demand	Point-to-Point Demand
Mixed	Point-to-Multipoint Reserved/Permanent	Point-to-Point Demand/Reserved
Permanent	Point-to-Multipoint Reserved/Permanent	Point-to-Multipoint Reserved/Permanent

Connection to the service is established on demand, reserved or permanent as required. The service should be accessible around the clock. It is user-to-controller, point-to-point (or point-to-multipoint, in some applications) and generally bi-directional asymmetric (or symmetric if no video monitoring function is used).

6.4.4.3 Networks

Ideally ISDN (typically one, two or three basic accesses of N-ISDN) or B-ISDN should be used for this application. The use of PSTN is possible although the performance would be very slow if video surveillance was used as part of the application.

6.4.4.4 Terminal Equipment

The most appropriate TE for this application is a PC platform. A dedicated terminal can be used but it would need the functionality of a PC.

6.4.4.5 Information interchange scenario

The site plan showing the position of all monitoring and controllable devices are stored at each site. The human controller interrogates all monitoring devices at the site and information flows from the control box into the human controller's terminal and is displayed. Based on the readings expected by reference to the stored information at the remote site (or situation as shown by video surveillance), the human controller might elect to change one, or more, of the settings at the site. In which case, control information is passed to the control box and the settings changed accordingly. The human controller can then interrogate the reading to check that the change in setting has been activated.

Another way of running the application is for the settings to be read on a rotation basis and displayed sequentially on the terminal. When the human controller sees a reading is not within the limits, control information can be passed to correct the situation.

6.4.4.6 Application configuration

In general:

- the terminal acts as an application and information generator;
- the control box and the associated store act as information providers with the control box also acting on the information (control information) supplied by the terminal.

The presentation objects are provided by the remote store and the terminal.

The TE is provided by the user, although the equipment itself may be bought, rented or leased. The control and monitor equipment is generally of special design although it may include standard equipment, such as cameras, ovens and refrigerators.

A simple configuration of a real-time control application is given in figure 10.

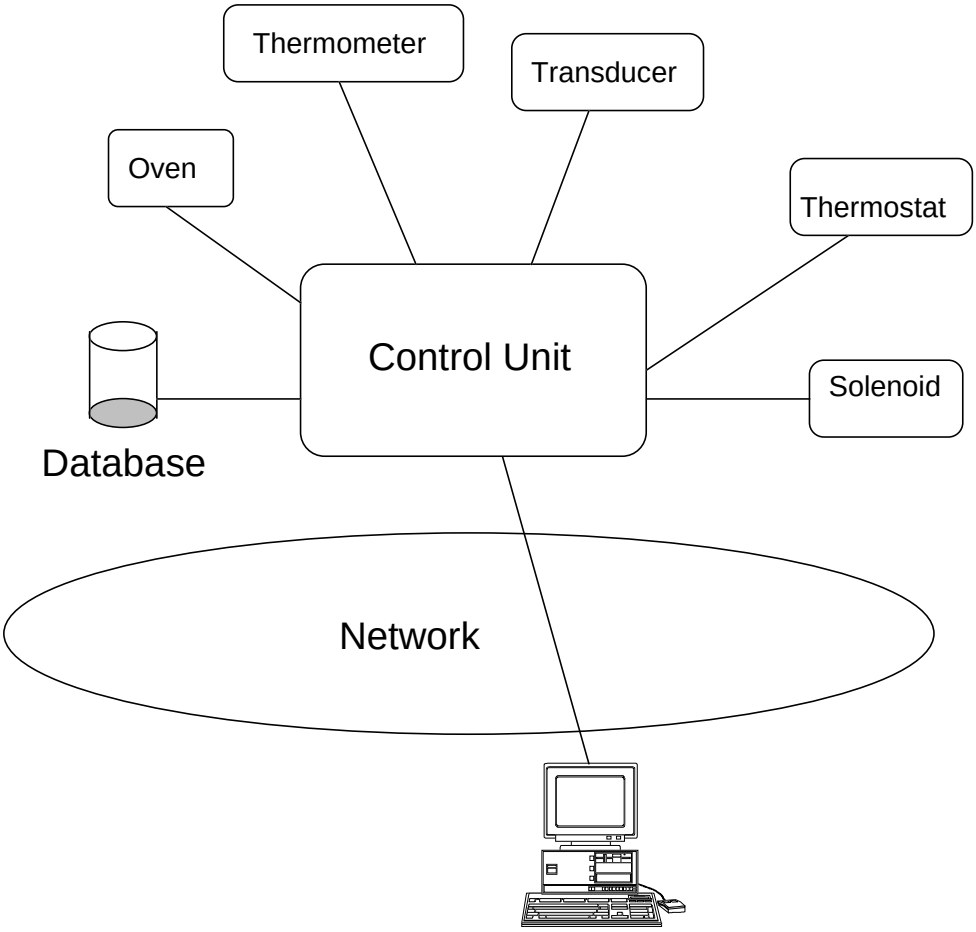

Figure 10: Application configuration

6.4.4.7 Use of MHEG

MHEG objects are interchanged bi-directionally between the control box store and the terminal via the access network.

MHEG objects express the presentation information, and the information of the monitoring devices.

6.4.4.8 Use of scripts

It is useful to interchange scripts:

- to allow external device control, e.g. ovens, cameras, door controls, VCR programming etc,...;
- to compute results of setting changes;
- to perform local simulations based on proposed changes in settings, etc,...

6.4.4.9 Application architecture

The application architecture corresponds to the terminal-to-terminal configuration presented in the general model.

6.4.4.10 API requirements

Table 24 gives an overview of the API functionalities that are required by the terminal in this particular application.

Table 24

Function family	Needed
Session	✓
Directory	
Interchange	✓
Accessor	✓
Modifier	✓
Handling	
Behaviour	
Exception	✓

6.4.4.11 SIR requirements

Table 25 gives an overview of the SIR functionalities that are required in this particular application.

Table 25

Function family	Needed
external device control	✓
external device control for data acquisition	✓
manipulation of MHEG objects	✓
access to external data	
access to external advanced calculation capability	✓
computations, variable handling and control structures	

7 Functional requirements on the MHEG API

This clause discusses and specifies the functions that the MHEG API standard should support in order to meet the requirements of the target M&H applications.

For this purpose, this clause:

- provides a synthesis of the application requirements as resulting from clause 6;
- presents a functional model and terminology applicable to the MHEG API;
- discusses and states the scope of the MHEG API from a functional perspective.

NOTE: There can be several MHEG engines in an application, all of them providing the MHEG API. In the following clauses, whenever the semantics of use of the MHEG API are discussed, "the MHEG engine" always refers to the particular MHEG engine that provides the MHEG API under use.

7.1 Synthesis of application requirements

This subclause provides a synthesis of the requirements of the target applications developed in clause 5, as regards the manipulation and interchange of MHEG objects. All these requirements are functional requirements.

7.1.1 Function targets and families

For the purpose of describing these functional requirements, the functions are clustered into families characterised by their target type (i.e. the kind of concept on which they apply) as well as the type of operation that applies to this concept. This set of families is believed to cover the whole range of common application needs.

The target types are the following:

- MHEG object;
- set of the MHEG objects;
- MHEG engine entity;
- set of the MHEG engine entities;
- the MHEG engine.

MHEG object is the target for functions that apply to one MHEG interchange object, i.e. an object which is encoded in MHEG format and currently available to the application, i.e. stored or interchanged in any accessible location.

Set of the MHEG objects is the target for functions that apply to the whole set or part of the set of all the objects which are encoded in MHEG format and currently available to the application, i.e. stored or interchanged in any accessible location.

MHEG engine entity is the target for functions that apply to one MHEG entity currently available to and under control of the MHEG engine, i.e. an MH-object, an rt-object or a channel handled by the MHEG engine.

Set of the MHEG engine entities is the target for functions that apply to the whole or part of the set of all MHEG entities handled by the MHEG engine, i.e. available MH-objects, rt-objects (including sockets) and/or channels.

The MHEG engine is the target for functions that apply to the MHEG engine globally.

The following function families are defined:

- session;
- directory;
- interchange;
- accessor;
- modifier;
- handling;
- behaviour;
- exception.

The **session** functions are related to the communication between MHEG using application and MHEG engine via the API. These functions are targeted at the MHEG engine. Examples are "open communication session" (possibly subsumes "start MHEG engine"), "close communication session" (possibly subsumes "stop MHEG engine"), get MHEG engine resource capabilities.

The **directory** functions deal with the management of MHEG objects and their availability to the application. These functions are targeted either at the set of MHEG objects or at a specific MHEG object. Examples are "list available objects", "select objects (matching certain criteria based for instance on attributes or location)", "add/remove object (to/from application availability)", "associate object with alias".

The **interchange** functions deal with the modification of the location of an MHEG object available to the application. These functions are targeted at an MHEG object. Examples are "send (or deliver) object", "retrieve (or pre-load) object", "store object".

The **accessor** functions allow to retrieve the value of an attribute of an MHEG object available to the application. These functions are targeted at an MHEG object. Examples are "get version number", "get MHEG ID", "get owner", "get content classification".

The **modifier** functions allow to modify (update) the value of an attribute of an MHEG object available to the application. These functions are targeted at an MHEG object. Examples are "set version number", "set MHEG ID", "set owner", "set content classification".

The **handling** functions deal with the handling of MHEG entities. They are targeted at the set of entities handled by the MHEG engine. Examples are "list open channels", "select available objects (matching certain criteria based for instance on status or type)".

The **behaviour** functions have semantics that match MHEG actions. They are targeted at an MHEG entity, i.e. an MH-object, an rt-object (including a socket) or a channel available to the MHEG engine. Examples are "get current speed", "set current speed", "run rt-object", "prepare/destroy MH-object", "new/delete rt-object", "return numeric/MH-object", "get availability status". Unlike the other ones that match MHEG actions being sent to MHEG entities, the return functions are designed to allow the MHEG using application to handle events generated by the MHEG engine after the triggering of a return action.

The **exception** functions allow to handle errors notified by the MHEG engine on the occurrence of an unexpected situation.

Table 26: Function families according to targets

Family/Target	MHEG engine	set of MHEG objects	MHEG object	set of MHEG engine entities	MHEG engine entity
Session	✓				
Directory		✓	✓		
Interchange			✓		
Accessor			✓		
Modifier			✓		
Handling				✓	
Behaviour					✓
Exception	✓				

7.1.2 Classification of application requirements

Not all applications need be provided with all families of functions. This is summarised in table 27.

Table 27: Application requirements for function families

Application/Family	Session		Directory		Interchange		Accessor	
	T	H	T	H	T	H	T	H
Encyclopaedia	✓		✓		✓		✓	
PoI/PoS	✓	✓	✓	✓	✓	✓	✓	✓
Telematic training	✓	✓		✓	✓	✓		✓
EPG	✓	✓	✓	✓	✓	✓		✓
Teleshopping	✓	✓	✓	✓	✓	✓	✓	✓
iVoD	✓	✓	✓	✓	✓	✓	✓	✓
Videoconference	✓				✓		✓	
Videotelephony	✓				✓		✓	
Telecontrol	✓				✓		✓	

Application/Family	Modifier		Handling		Behaviour		Exception	
	T	H	T	H	T	H	T	H
Encyclopaedia			✓		✓		✓	
PoI/PoS	✓	✓	✓	✓	✓	✓	✓	✓
Telematic training		✓		✓	✓		✓	✓
EPG	✓	✓		✓	✓		✓	✓
Teleshopping	✓	✓	✓	✓	✓		✓	✓
iVoD	✓	✓	✓	✓	✓		✓	✓
Videoconference	✓				✓		✓	
Videotelephony	✓		✓		✓		✓	
Telecontrol	✓						✓	
Abbreviations: T: Terminal H: Host								

7.2 Model and terminology

This subclause describes the terminology applicable to the API, and illustrates the different levels of an API using an indicative diagram.

7.2.1 Terminology

An API consists of **primitives**, i.e. basic entry points provided by a **provider** module to any **user** module to enable the user to access **software services** supplied by the provider. These modules are pieces of software, although they can use services provided by computer hardware or other electronic equipment.

In the present case, the **MHEG API** gives access to **MHEG object manipulation** services. **MHEG engines** are the providers, whereas **MHEG applications** are the users. One MHEG engine may provide its software services to several applications. An MHEG engine can therefore be viewed as a **server**, whereas MHEG applications are the **clients** of the MHEG object manipulation service.

An API primitive is used to transfer some information between its user and its provider. This information consists of control and/or data. The information may be forwarded either from a client to its server or from the server to one of its clients. The information may be generated by its sender either on its own initiative or as a reply to a formerly issued primitive. The following terms are used:

- a **request** is a primitive issued by the client on its own initiative to forward information to the server;

- a **response** is a primitive issued by the server as a reply to a request to forward information to the client;

- a **notification** is a primitive issued by the server on its own initiative to forward information to the server.

Different kinds of requests may be considered:

a) requests that require no response are called **asynchronous requests**;

b) requests that do not require an immediate response are called **deferred synchronous requests**;

c) requests that require an immediate response, until which the client process cannot proceed, are called **synchronous requests**.

Whether synchronous or asynchronous, requests may result in processing that will in turn trigger notifications.

7.2.2 Levels of abstraction of the API

The concept of "Level of abstraction" is complex with several (possibly non-conflicting) uses. Usage of "Level of abstraction" implies variation in the amount of functionality offered to the calling program by each invocation.

The same service may be provided by multiple API specifications which differ in level of abstraction (see example).

EXAMPLE: A less abstract (primitive level) API specification for MHEG may provide the application programmer low level primitives that, if applied in the right sequence, result in a dynamic effect on the screen. On the other hand a more abstract API specification (function level) may provide a simple, single subroutine call providing this very specific animation effect.

More abstract (higher level) APIs are closer to application semantics, but less abstract APIs (lower level) are more "primitive" and therefore fit more applications requirements (hence allowing more application portability) with a smaller number of primitives.

Three levels of abstraction can be identified:

- the function level;
- the primitive level;
- the message level.

Only the part for use by a client software application is considered, although it is likely to be implemented in a symmetric way on the server (MHEG engine) side.

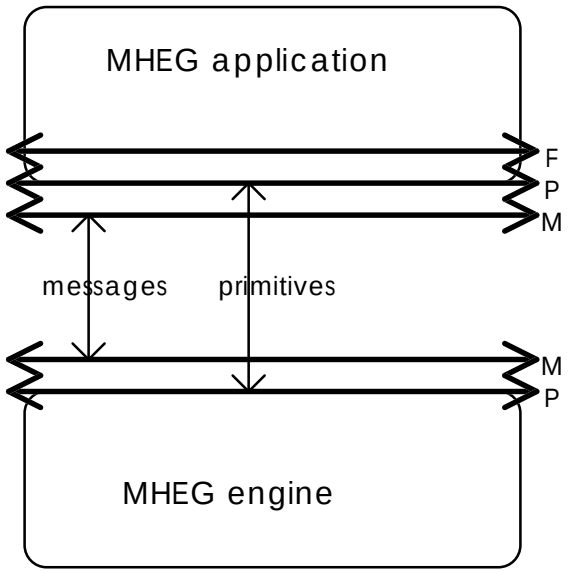

Figure 11: Communication between MHEG application and MHEG engine

At the **function level**, the software application needs to access the MHEG object manipulation and interchange functions in the most practical way. If this optional level exists, it aims at providing the application designer with a high-level, application-oriented representation of the MHEG object manipulation and interchange functions, as matching the expressed requirements. This level is application-dependent and language-dependent. For instance, if the applications are to be written in C++, they might make use of a C++ class library whose classes match application-oriented concepts and whose public methods encapsulate primitive level calls to provide MHEG object manipulation and interchange functions. Such a library might provide for example a scene object class (provided the application uses such concepts) for representing the particular MHEG composite rt-objects handled by the application. This class would provide public methods for creating, modifying and presenting the scene. These functions would in turn call such API requests as "new rt-composite", "set rt-composite attributes" and "run rt-composite" to perform the corresponding tasks. The library might also associate each object class with an event handler to deal easily with the functions of the return family. The function level is defined by the application-oriented semantics of the functions.

The **primitive level** consists of basic, low-level, MHEG-representation-oriented entry points of the MHEG engine as defined by the future MHEG API standard. It corresponds to an implementation of the API on a given platform, using a given programming language. This level is language-dependent but application-independent and system-independent. It will usually match one-to-one API primitives with procedures, subroutines, functions, methods, messages or whatever mechanism is provided by the programming language. This implementation is therefore dependent on a programming language binding, i.e. a way of mapping API primitives to programming language constructs. The primitive level is defined by the syntax and MHEG-oriented semantics of the API primitives.

At the **message level**, the information forwarded by the API primitives need be interchanged between software modules in a system-dependent way. The interchanged data are called messages. The message encoding will usually be system-dependent and possibly language-dependent, but application-independent. The message level is defined by the coded representation of the messages according to the primitives syntax.

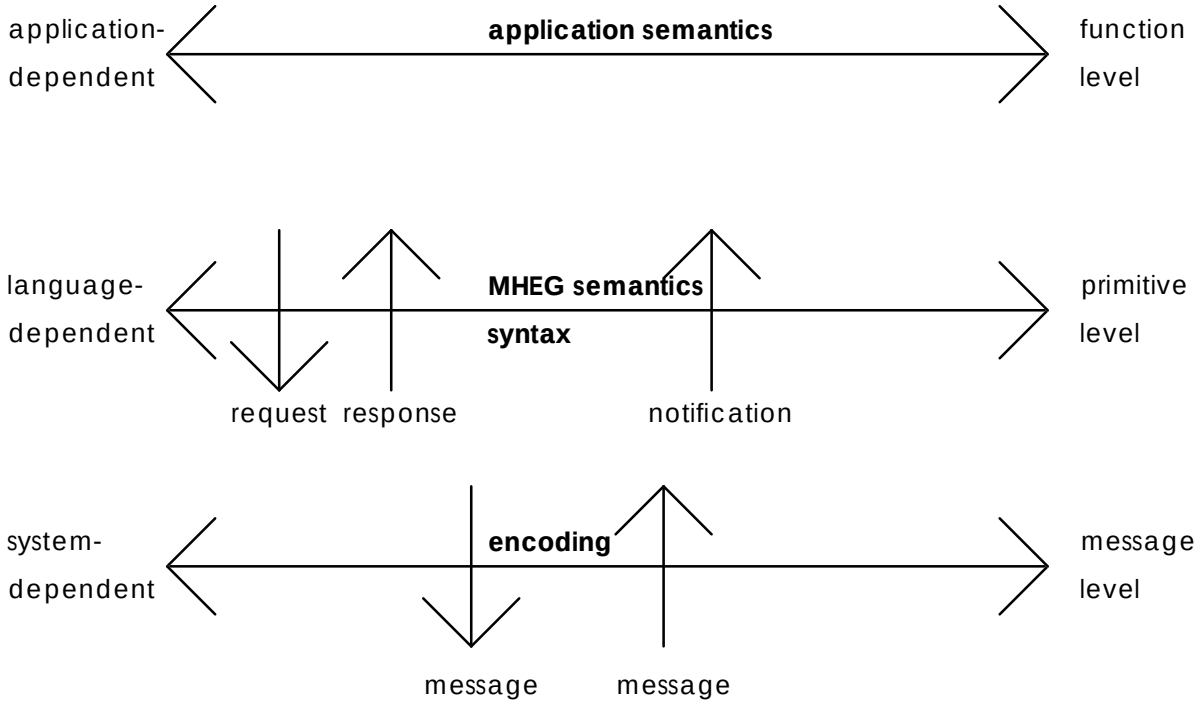

Figure 12: Levels of implementation of the MHEG API

The guideline for selection of the right level of abstraction for the API is to stay as low-level as possible as long as it remains platform-independent.

7.3 Functional scope of MHEG API

The MHEG API is the interface that any MHEG engine conforming to the MHEG API standard should provide to MHEG applications. The scope of the MHEG API is to provide applications with functions for the manipulation and interchange of MHEG objects. This subclause discusses and recommends which function families should actually be specified in the MHEG API standard and required from MHEG engines.

There are two questions arising from the requirements summarised in subclause 7.1:

- how do the application-required functions map to the MHEG API primitives?
- what functions should the MHEG API provide among the listed families?

7.3.1 Mapping application functional requirements to MHEG API primitives

Application designers are likely to base themselves on high-level services provided by the above described function level. Which services should be provided, in terms of which concept abstractions such services should be defined, is likely to be dependent on the application domain. Whether such services should be standardized is an open issue.

The MHEG API should lie at the lowest, most basic level of functional abstraction. The semantics of the MHEG API primitives should therefore be MHEG-oriented rather than application-oriented. For instance, most primitives of the behaviour function family would match MHEG elementary actions. The MHEG API should support any application requirement for MHEG object manipulation and interchange, so that a higher-level interface could always be built upon the API without any necessary modification of the MHEG engine interface. A function should not be provided as a primitive if it can be expressed in terms of other, more basic primitives.

Functions that might optionally be provided by MHEG engines to ease the design or maintenance of client software applications, but are not required for such applications to run, should not be required from an MHEG engine and are therefore considered as outside the scope of the future standard. The following functions belong to this excluded category:

- monitoring of the use of MHEG objects by an MHEG engine;

- history of the handling of MHEG objects by an MHEG engine;
- browsing through the MHEG objects currently handled by an MHEG engine.

7.3.2 Functions to be provided by the MHEG API

ISO/IEC DIS 13522-1 [2] does not standardize the MHEG engine functionality, but nevertheless makes some assumptions on what functions an MHEG engine should provide. The MHEG engine is responsible for the handling of interchanged MHEG objects. The responsibility for the interchange of MHEG objects is not clearly allocated to the MHEG engine nor to any other defined module.

With respect to the reference model developed in clause 4, it can be assumed that the MHEG object interchange functionality will be performed by the access agent, which provides services to MHEG engines and other modules, including MHEG applications.

ISO/IEC DIS 13522-1 [2] proposes an indicative list of the functions that could be provided by an MHEG engine as an interface to its using application. This description assumes that the MHEG engine is responsible only for the processing of MHEG objects and its interactions with the using applications.

The question therefore arises as to whether the MHEG API should provide the following functions families (see subclause 7.1):

- directory;
- interchange;
- accessor;
- modifier.

Indeed, it could be argued that:

- accessor and modifier function families might be provided by the application itself, provided it integrates an MHEG parser;

- interchange and directory function families might be provided by the access agent.

As concerns accessor and modifier function families, integrating an MHEG parser in every application would not be an optimal solution in any case. Since the MHEG engine is able to parse MHEG objects, it would be a lot easier if it could provide the associated services through the MHEG API.

As concerns interchange and directory function families, providing them through the MHEG API could facilitate their use by the application, especially for the host application as real-time sending of MHEG objects is involved. Moreover, such functions are part of the scope of the "API for manipulation and interchange of M&H information".

On another hand, the semantics of these functions are highly dependent on the application architecture as concerns where and how the objects can be stored, interchanged or managed in a directory. Such location and transfer mode information is under the control of the access agent.

The best solution for those directory and interchange functions which depend on the location and transfer mode is therefore as follows:

- these functions have a default trivial behaviour which is determined by the MHEG engine, e.g. store object stores it on the local disk;

- these functions may be called with one optional parameter (of undetermined type) which is a symbolic identifier provided by the access agent and representing the location, transfer channel or whatever allows to identify where and how storage, interchange or directory management takes place.

As a conclusion, all the function families described in subclause 7.1 should be provided by the MHEG API.

8 Technical requirements on the MHEG API

This clause addresses the technical aspects of the MHEG API standard definition.

For this purpose, this clause:

- analyses general API specification guidelines;
- states the technical requirements on the MHEG API and their consequences;
- discusses technical options allowing to meet these technical requirements.

8.1 Guidelines for API specification

ISO/IEC JTC1 N 2965 recommends guidelines for API standardization. It first provides a definition of what an API is, as follows:

> "An API is a boundary across which application software uses facilities of programming languages to invoke services. These facilities may include procedures or operations, shared data objects and resolution of identifiers. A wide range of services may be required at an API to support applications. Different methods may be appropriate for documenting API specifications for different types of services.
>
> The information flow across the API boundary is defined by the syntax and semantics of a particular programming language, such that the user of that language may access the services provided by the application platform into the syntax and semantics of the programming language.
>
> An API specification documents a service and/or service access method that is available at an interface between the application and an application platform."

Following the ISO/IEC JTC1 N 2965, an API may be specified as:

- the description of a programming language;

- a language-specific API specification, i.e. a description of the semantics of a set of functionality in the syntax and data types of an existing programming language;

- a language-independent API specification, i.e. a description of the semantics of a set of functionality in an abstract syntax using abstract data types.

The latter is also known as an **abstract API specification**. The advantage of such a specification is that it serves as a reference to assure consistency across several language bindings. Indeed, the primary function of an API in the multimedia platform definition is to support application portability (as given in ISO/IEC JTC1/SC18/WG1 N 1632).

The abstract API specification is therefore the most appropriate one to define an API which should be made available to client software that may use any programming language, as recommended by ISO/IEC JTC1 N 2965:

> "Where a standardization project for an API specification includes multiple language bindings with common interface characteristics, the use of language-independent API specification should be strongly encouraged".

8.2 Technical requirements on the MHEG API specification

The MHEG API standard should meet the following requirements:

- portability;
- genericity;
- conformance testability;
- implementability.

8.2.1 Portability

The portability requirement can be expressed as follows: the MHEG API standard should enable MHEG applications to use the MHEG object manipulation and interchange service provided by MHEG engines in a way independent of:

- the programming language used for the MHEG application;

- the underlying operating system.

To meet the portability requirement, as stated in subclause 8.1, it is recommended to develop an abstract API specification, i.e. a language-independent API specification describing the semantics of the API functionality in an abstract syntax using abstract data types. This definition should be independent of both:

- the mechanism used for interchanging information between the API user and the API provider, i.e. the messages that are exchanged as the result of triggering API primitives;

- the actual encoding of these messages.

8.2.2 Genericity

The genericity requirement can be expressed as follows: the MHEG API standard should provide appropriate support to cover all the common requirements of MHEG applications.

To meet the genericity requirement, as discussed in subclause 7.3, it is recommended that the MHEG API should lie at the most basic level. As far as possible, the primitives defined by the MHEG API should match MHEG elementary actions and the data types used by the MHEG API should match MHEG data types. This guarantees to maximise the range of MHEG object manipulations made available to applications.

8.2.3 Conformance testability

The conformance testability requirement can be expressed as follows: the MHEG API standard should make it as easy as possible to ensure the conformance of MHEG engines to the MHEG API standard, i.e. the correct provision of this API by an MHEG engine under test, and also possibly (subject to discussion) the conformance of MHEG applications to the MHEG API standard, i.e. the correct use of this API by an MHEG application under test.

To meet the conformance testability requirement, it is recommended that the ETSI methodology be followed and that conformance testing standards be used whenever applicable.

The methodology recommends the following approach:

1) state clear static conformance requirements;
2) state clear dynamic conformance requirements;
3) specify conformance testing procedures allowing to implement these test purposes.

ETSI standardization processes make use of standards (especially ISO/IEC 9646 Parts 1 to 5 [19]) for the conformance testing specification of protocols. There is currently no standard for the conformance testing specification on APIs. Whether the currently used standards can apply to APIs is an unresolved issue. For the MHEG API, the statement of conformance requirements and test purposes will allow to decide upon a methodology for conformance testing specification.

Conformance testing is discussed in subclause 8.2.7.

8.2.4 Implementability

The implementability requirement can be expressed as follows: the MHEG API standard should take into account simplicity and clarity both in the definition and the formulation to make implementation of conforming MHEG engines as easy as possible.

To achieve the implementability requirement, it is recommended that both language bindings and message encoding rules be easy to deduce from the abstract API specification. This is discussed in the subclauses 8.2.5 and 8.2.6.

8.2.5 Language bindings

To achieve portability, an abstract API specification should be developed. However, to achieve implementability, language bindings should be easy to deduce from this abstract specification.

Among recommended policies of ISO/IEC JTC1 N 2965 revised (05/94):

> "The use of common, standardized methods where available for the specification of language-independent API specification should be encouraged".
>
> "Where a standardization project for an API specification includes a language-independent API specification, the language-independent API specification shall be progressed together with at least one language binding that depends on the language-independent API specification".

Possible language bindings to be progressed together with the MHEG API specification include the following languages, which are the most likely to be used for programming MHEG applications:

- binding with a widespread, preferably object-oriented, programming language; C++ would appear the best choice;

- binding with a scripting language; this imposes the choice of a proprietary scripting language;

- binding with the future standard SIR; this relates it to the script representation work item.

8.2.6 Message encoding

Messages refer to the structure (syntax) and encoding of the bits of information which are interchanged both ways between implementations of the API user and the API provider considered as independent systems.

In a number of cases, the actual structure and encoding of the messages does not matter as long as they are understood by both the API user and the API provider. This happens in most cases where the API user (MHEG application) and the API provider (MHEG engine) both are pieces of software running on the same computer system, e.g. on the terminal. In this event, whoever provides an MHEG engine for public use (e.g. a software vendor) may for instance provide with it a set of client libraries to be linked with any MHEG application, so that every library enables the application to access the MHEG API in one particular programming language, under a particular operating system. There could for instance be an MHEG API client library for C++ under Macintosh, one for C++ under Windows, one for Visual Basic under Windows, each of them implementing the MHEG API by an MHEG application written in that particular language on this particular system, without care for the actual way messages are encoded and interchanged.

In some cases however, being able to encode messages in a standard way and knowing how to do it may be a relevant issue. This is especially the case wherever the MHEG API has to be used by an application that does or may not run on the same computer as the MHEG engine. Since the primary objective of this standardization being performed by ETSI is to allow the use of MHEG by telecommunication applications, such distributed configurations should not be excluded. In such cases, underlying operating systems may differ and such problems as byte ordering have to be taken into account.

The MHEG API standard should then recommend at least one standard-based method for encoding messages that are interchanged between MHEG applications and MHEG engines as the result of triggering API primitives.

8.2.7 Conformance testing

Among recommended policies of ISO/IEC JTC1 N 2965 revised (05/94):

> "API conformance requirements should include sufficient level of specification that verification test methods can be readily derived. .....The use of API specification methods that support the use of automated test procedures should be encouraged".

All implementations of the MHEG API need to be tested to determine whether they conform to the MHEG API specification that results from the work of this project team. It would be ideal if standardized abstract test suites could be developed for the MHEG API for use; by suppliers or implementors in self-testing; by telecommunications administrations or PNOs; or by third party testing organizations. This would lead to compatibility and wide acceptance of test results produced by different test laboratories, and thereby minimise the need for repeated conformance testing of the same implementation.

The standardization of test suites has been defined and there exists a common testing methodology, together with appropriate testing methods and procedures, within document ISO/IEC 9646 Parts 1 to 5 [19] and ETR 141 [20]. The document is applicable to the different phases of the conformance testing procedure, these phases being characterised by three major activities, as follows:

- the specification of abstract test suites for particular Open Systems Interconnection (OSI) protocols;

- realisation of the means of executing specific test suites;

- the conformance assessment process carried out by a test laboratory culminating in the production of a Protocol Conformance Test Report (PCTR), which gives the results in terms of the protocol specification and test suite used.

To evaluate the conformance of a particular implementation, it is necessary to have a statement of the capabilities and options which have been implemented, for the relevant protocol, so that the implementation can be tested for conformance against those requirements only. Such a statement is called a Protocol Implementation Conformance Statement (PICS). In a PICS there should be a distinction between the following categories of information:

a) information related to the mandatory, optional and conditional static conformance requirements of the protocol itself;

b) information related to the mandatory, optional and conditional static conformance requirements for multi-layer dependencies.

If a set of interrelated protocols has been implemented in a system, a PICS is needed for each protocol. A System Conformance Statement will also be necessary, itemising all protocols in the system for which distinct PICS is provided.

The test notation recommended to be used for telecommunications is the Tree and Tabular Combined Notation (TTCN). TTCN is designed to meet the following objectives:

a) to provide a notation in which abstract test cases can be expressed in standardized test suites;

b) to provide a notation which is independent of test methods, layers and protocols;

c) to provide a notation which reflects the abstract testing methodology defines in ISO/IEC 9646 Parts 1 to 5 [19].

Due to the generic nature of the MHEG API being defined by the project team coupled, and the fact that a standardized protocol does not exist for the MHEG API, the above methodology cannot be used. There is another conformance testing methodology that can be used in these circumstances, and this involves the use of a PIXIT and other Points of Control and Observation (PCOs). With this method, the test laboratory will require information relating to the Implementation Under Test (IUT). The IUT, in this case, is the MHEG API interface whereas the System Under Test (SUT) is the system containing the MHEG Engine, the application and other software and hardware, as appropriate to the particular implementation. The implementor will complete a Protocol Implementation eXtra Information for Testing (PIXIT) proforma from which it should be possible to stimulate, control and monitor the MHEG API interface. This will involve

applying stimuli at other interfaces within the SUT to cause an action at the MHEG API interface. This method, however, whilst it can be used by individual implementor for in-house testing, is not suitable for conformance testing purposes as the other interfaces of the SUT are not standardized and can vary from being other software modules to being interfaces to television cameras. It is obvious that this would involve test houses in acquiring vast amounts of test equipment to cope with the variety of interfaces that they would ever encounter.

Due to a number of factors: that the MHEG API definition will be generic; events at the interface are not predictable and no standardized protocol is being defined; that the implementation of the **Application** and **MHEG Engine** could be such that the MHEG API interface is not indistinguishable; and that the project team do not want to constrain implementations, there is a high possibility that under certain sets of conditions that no conformance testing is possible.

The only way in which conformance testing is possible is that under test conditions the **Application** is replaced by test software and the MHEG API is designed so that some form of predictability is added (such as an acknowledgement primitive in response to a request). The merits of including special primitives for test purposes only (either switched on during testing or as part of run-time) can be disputed and is for further study. For those cases where testing is possible, it will be necessary to write generic test purposes in plain language for the MHEG API interface. These can then be converted into TTCN or C/C++ etc,... depending on the implementation. The selection of the tests can be then based on the completion of an Implementation Conformance Statement (ICS) by the implementor. Future work in this area is dependent on liaisons with the appropriate groups within ETSI, EWOS, ISO/IEC/JTC 1/SC21 etc,...

Further information on abstract test suites, ICS, PICS, PIXIT and TTCN can be found in ISO/IEC 9646 Parts 1 to 5 [19] and its referenced documents.

8.3 Technical options

This subclause discusses technical choices allowing to meet the technical requirements on the MHEG API definition as expressed and discussed in subclause 8.2.

To follow the recommendations expressed in subclauses 8.1 and 8.2.5, abstract API specification methods, preferably standardized ones, should be identified; selection among them would then occur according to their allowing to meet the technical requirements.

ETSI has up to now addressed specification and conformance testing methods for protocols, as well as general architectures for PCIs; however, specification of high-level APIs is rather a new dimension of ETSI work, especially brought in the framework of specifying a core multimedia toolbox; API specification methodologies has therefore not yet been addressed.

8.3.1 IEEE OSI abstract data manipulation

ISO/IEC JTC1/SC21 is responsible for the specification of APIs within ISO. General guidelines are under development (ISO/IEC JTC1 N 2965). However, no standard abstract API specification method has been standardized. Current standards or draft standards for abstract API specifications in the OSI world (see IEEE Std 1224.2-1993 [21]) use a method is defined by IEEE Std 1224-1993 [22].

This methodology is based on an object model. Operations are provided as the interface of classes which have attributes and are hierarchically organized in an inheritance tree.

The abstract API specification uses plain text specification, with each API primitive being described by:

- a synopsis (name of operation, status data type, name and type of input and output parameters);
- a description of the operation semantics;
- semantics of input arguments;
- semantics of output arguments;
- semantics of results;
- errors that may be returned;
- related (cross-referenced) primitives.

This methodology for abstract API specification is only intended as a standard for the OSI context. However, it could be adapted to any API specification, including the MHEG API. It allows to meet the technical requirements as follows:

- portability: the method provides a language-independent specification;

- genericity: It has no influence on the genericity requirement;

- conformance testing: the above mentioned standards are associated with conformance testing specification standards such as IEEE Std 1326-1993 [23];

- implementability: the above mentioned standards are associated with corresponding language binding standards.

This methodology allows to apply common sense procedures in a standard way. Its main drawback is that it does not provide any formal description technique for specifying the MHEG API in an abstract way. Practically, it therefore does not provide any significant added value as compared to using no standard at all.

8.3.2 OMG CORBA Interface Definition Language

The Object Management Group (OMG) and X/Open are non-profit, world-wide organizations aiming at developing public common specifications and encouraging market uptake of object technology (see document published by the Object Management Group: "Object management architecture guide") and open systems respectively. As an important milestone towards this objective, they have issued the Common Object Request Broker Architecture (CORBA) specification (see document published by X/Open and the Object Management Group: "The common object request broker: Architecture and specification") of an open architecture enabling objects to interoperate by providing and using services in distributed environments.

One part of this specification is the Interface Definition Language (IDL). This language is a formal description technique for specifying the services provided by objects for use by applications or other objects. IDL can be used independently from the CORBA context. Although object-oriented communication in distributed environments is actually a technology of some relevance with regard to the definition of a multimedia core toolbox, this report only considers the use of IDL and its underlying object model as a context-independent formal description technique for the specification of APIs.

Application of IDL should be based on an underlying object model. Such an object model is defined in terms of **object types** which support **operations** characterising the behaviour of objects. **Objects** are instances of object types. Objects may be identified using **object references**. **Non-object types** can be instantiated but do not support operations. Operations are defined by a **signature** consisting of a name, a list of input or output **parameter** types and a list of **result** types. The set of operation signatures defined for a type is the **interface** of that type. **Subtyping** allows to define type hierarchies, with subtypes providing their supertypes' interface as a part of their own interface. **Operation requests** may have different operational semantics such as synchronous, asynchronous, etc,... Consequences of an operation request include side effects, results and **exceptions**.

IDL is the language used to describe the interfaces (i.e. the set of operations) provided by objects. It consists of lexical conventions, preprocessing directives and an Extended Backus Naur Form (EBNF) grammar. An IDL specification of an API consists of data type definitions, constant definitions, exception definitions, interface definitions and module definitions. More detailed examples are developed in clause 9.

IDL is currently used within ISO for the specification of distributed multimedia environments (see ISO/IEC CD 14478-1 [24]).

8.3.3 Use of IDL for the MHEG API definition

As shown in clause 7, the MHEG engine interface consists of a set of API primitives that can be organized into clusters according to which entity provides the interface. Definition of this interface can therefore logically be structured according to the operation provider. In the MHEG API context, objects that provide interfaces need not be implemented as separate object implementations. More likely, they would be internal entities handled by the MHEG engine. As for the MHEG standard, the MHEG API should then

follow an object-oriented definition methodology without requiring the implementations to use object-oriented design or programming techniques.

This technique is quite similar to the technique mentioned in subclause 8.3.1. It provides at least functionally equivalent features. It however provides the following additional features affecting implementability and portability:

- language binding: guidelines for C language binding of IDL specifications are part of the IDL specifications; due to the IDL syntax, C++ language binding is also very easy; other language binding guidelines could be developed in the same way;

- formal description: unlike the technique mentioned in subclause 8.3.1, IDL provides a complete formal description language which allows a very concise, readable and efficient specification of the MHEG API. Moreover, this formal description language is also appropriate for automatic compilation, which means that MHEG API implementations could be automatically generated for a given language and operating system using appropriate IDL compilers. This of course could be a major element in facilitating implementability and general use of the standard API rather than any specific interface.

8.3.4 Use of ASN.1

Abstract Syntax Notation One (ASN.1) is basically a standard notation that allows the definition of the structure and encoding of data. ASN.1 guarantees the language independence and platform independence at both the syntax description and encoding levels. The provided base types are close to those used in programming languages.

As a technique for defining data types, ASN.1 cannot therefore be considered as an appropriate candidate for the definition of the interface. On the other hand, ASN.1 is appropriate for the definition of the messages, i.e. the data interchanged between the API user and the API provider.

It is therefore recommended that ASN.1 be used as the method for describing the structure and encoding of the MHEG API messages, whenever a standard encoding of these messages is required. However, the conformance requirements on the MHEG API would be only at the primitive, not at the message level. Any MHEG API implementation could therefore decide whether to provide this base encoding or not, according to the type of configuration it is expected to function.

Another reason for considering the use of ASN.1 is that ASN.1 is the base coded representation for MHEG objects. This means that the structure of MHEG object and data types and the encoding of MHEG objects is defined using ASN.1. Therefore, since most of the data types which will be manipulated by the API match MHEG data types, use of ASN.1 for the message encoding allows to maximise coding efficiency, to re-use elements of the MHEG standard specification instead of developing new ones and to minimise data transcoding requirements from both the MHEG engine and the MHEG applications.

The use of ASN.1 as the message encoding technique has still another application which is the expression of MHEG API requests in the SIR formulation. In the event that ASN.1 would be chosen as the base notation for the SIR, invocation of MHEG API requests could be encoded exactly in the same way as API underlying messages.

8.3.5 Recommendations

As a conclusion, it is recommended that:

- an IDL specification be developed as the normative specification of the MHEG API primitives;

- an ASN.1 specification of the underlying messages be developed as an informative part of the MHEG API standard.

Since IDL is not a recognised international standard, the formal description of the Interface Definition Language used should be included as a normative part of the MHEG API standard.

9 Methodology for the specification of the MHEG API

This clause presents the recommended methodology for the specification of the MHEG API.

9.1 Summary of methodology

To meet the functional requirements and technical requirements presented in clauses 6 and 7, in accordance with the conclusions presented there, the MHEG API standard drafting activity to occur in the next phase should proceed according to the following, consecutive steps:

1) (normative) state conformance requirements;

2) (normative) develop the MHEG API object model;

3) (normative) define the complete interface definition language syntax and semantics used for the MHEG API;

4) (normative) develop an IDL specification of the MHEG API;

5) (informative) develop an ASN.1 specification of the messages underlying the MHEG API;

6) (informative) express guidelines and mechanisms for specifying the language bindings of the MHEG API, optionally developing the C++ language binding as an example;

7) (informative) express static and dynamic conformance testing purposes, optionally suggesting guidelines for the derivation of conformance testing procedures from the API specification.

The following should be subject to further standardization work:

- language bindings for one or several common languages, including the SIR;

- conformance testing procedures for both MHEG engines providing the MHEG API and MHEG applications using the MHEG API.

9.2 Object-oriented analysis of the MHEG API

The object-oriented analysis of the MHEG API is based on an analysis of the MHEG standard. This standard is currently at DIS proposal level and therefore still subject to change. The results of this subclause should therefore be reviewed during the MHEG API specification according to possible changes in ISO/IEC DIS 13522-1 [2].

9.2.1 Object types

It should be remembered that the objects described hereafter are introduced as useful concepts for specifying the interface, but are not required to be implemented as separate objects. The MHEG API is specified as an abstract API in terms of operations provided by objects, but implementations of the MHEG API will be provided by MHEG engine implementations.

NOTE 1: Appropriate naming rules (including prefixing of operation, type or instance names) should be defined and followed throughout the MHEG API definition.

The base object types are the following:

- MHEGEngine;
- MHEGObjectManager;
- MHEGObject;
- EntityManager;
- Entity.

The MHEGEngine object type interface provides the operations related to the communication between the MHEG application and the MHEG engine. These operations match the session functions as described in subclause 7.1.

The MHEGObjectManager object type interface provides the operations related to the management and access to the directory of the MHEG objects available to the application. These operations match those directory functions described in subclause 7.1 that are targeted at the set of available MHEG objects.

The MHEGObject object type interface provides the operations related to the access and interchange of a single MHEG object. These operations match the interchange, accessor and modifier functions described in subclause 7.1, as well as those directory functions described in subclause 7.1 that are targeted at a single available MHEG object. MHEGObject is an abstract type (it cannot have instances) which has subtypes whose hierarchy follows the hierarchy defined by the MHEG standard, as defined by ISO/IEC DIS 13522-1 [2], subclause 16.1:

- MHEGAction;
- MHEGLink;
- MHEGModel (abstract);
- MHEGContainer;
- MHEGDescriptor;
- MHEGScript (direct subtype of MHEGModel);
- MHEGComponent (direct subtype of MHEGModel) (abstract);
- MHEGComposite (direct subtype of MHEGComponent);
- MHEGContent (direct subtype of MHEGComponent);
- MHEGMultiplexedContent (direct subtype of MHEGContent).

NOTE 2: MHEGObjects (and their subtypes) match form a) objects as defined in ISO/IEC DIS 13522-1 [2], subclause 6.2.4, i.e. objects available to the application.

The EntityManager object type interface provides the operations related to the handling of MHEG entities (mh-objects, rt-objects and channels) by the MHEG engine. These operations match the handling functions as described in subclause 7.1.

The Entity object type interface provides the operations related to the handling of the behaviour of a single MHEG entity. These operations match the behaviour functions as described in subclause 7.1. Most of them match MHEG elementary actions. Entity is an abstract type (it cannot have instances) which has subtypes whose hierarchy follows the hierarchy defined by the MHEG standard, as defined by ISO/IEC DIS 13522-1 [2], subclause 16.1:

- mhObject (abstract);
- rtObject (abstract);
- Channel;
- mhAction (direct subtype of mhObject);
- mhLink (direct subtype of mhObject);
- mhModel (direct subtype of mhObject) (abstract);
- mhContainer (direct subtype of mhObject);
- mhDescriptor (direct subtype of mhObject);
- mhScript (direct subtype of mhModel);
- mhComponent (direct subtype of mhModel) (abstract);
- mhComposite (direct subtype of mhComponent);
- mhContent (direct subtype of mhComponent);
- mhMultiplexedContent (direct subtype of mhContent);
- rtScript (direct subtype of rtObject);
- rtComponent (direct subtype of rtObject) (abstract);
- rtComposite (direct subtype of rtComponent);
- rtSocket (direct subtype of rtComponent);
- rtContent (direct subtype of rtComponent);
- rtMultiplexedContent (direct subtype of rtContent).

NOTE 3: mhObjects (and their subtypes) match form b) objects as defined in ISO/IEC DIS 13522-1 [2], subclause 6.2.4, i.e. objects available to the MHEG engine.

NOTE 4: rtObjects (and their subtypes) match form c) objects as defined in ISO/IEC DIS 13522-1 [2], subclause 6.2.4, i.e. instances of mhObjects available to the presentation process.

The exception functions as described in subclause 7.1 are provided as part of the interface of all objects.

9.2.2 Non-object types

Non-object types are simple or structured types used for expressing the interfaces. Most types are used to match data types used by the MHEG standard.

Simple non-object types consist of the simple types defined by IDL:

- numeric types: short, long, unsigned short, unsigned long, float, double;
- character types: char, string;
- boolean;
- octet;
- any.

Structured types can be constructed using the sequence, struct, union, array and enum constructs. ASN.1-IDL data type mapping would help matching MHEG data types to MHEG API data types. Examples of structured non-object types are:

- generic hook;
- generic reference;
- generic list;
- classification.

9.2.3 Analysis of behaviour functions

The behaviour functions have the following specific features:

- they are provided by a number of different objects;
- they are much more numerous than the other function families;
- they correspond to the actions described in the MHEG standard.

The behaviour function sub-families are summarised in table 28.

Table 28: Behaviour function sub-families

Behaviour sub-family	Interface provider object	Corresponding MHEG actions
Postponed	rtComponent Channel	Delay
Returnability	not applicable (event)	Return
Alias	Entity	Set Alias
Availability	mhObject	Prepare Destroy Get Preparation Status
Firability	mhLink	(to be provided by MHEG DIS)
Abort	mhLink	Abort (to be provided by MHEG DIS)
Generic value storage	mhContent	Get/Set Data
Copy	mhContent	Copy (to be provided by MHEG DIS)
Multiplexing/demultiplexing	mhMultiplexedContent	Set Multiplex Set Demultiplex
Availability	rtObject	New Delete Get Availability Status
Running	rtObject	Run Stop Get Running Status
Termination	rtScript	Get Termination Status (to be provided by MHEG DIS)
Passing parameters	rtScript	Set Parameters (to be provided by MHEG DIS)
Presentation and structural dynamism	rtSocket	Plug
Channel assignment	rtComponent	Get/Set Channel Assignment
Perceptability	rtComponent	Get/Set Opacity Get/Set Presentation Priority
Temporal	rtComponent	Get/Set Visible Duration Get/Set Current Temporal Position Get/Set Visible Duration Position Get/Set Speed Get/Set Timestones Get/Set Timestone Status
Audible	rtComponent	(to be provided by MHEG DIS)
Stream choice	rtComponent	(to be provided by MHEG DIS)
Spatial	rtComponent	(to be provided by MHEG DIS)
Channel availability	Channel	New Delete Get Availability Status
Channel perceptability	Channel	Get/Set Perceptability
Selection	rtComponent	(to be provided by MHEG DIS)
Modification	rtComponent	(to be provided by MHEG DIS)
Interaction style	rtComponent	(to be provided by MHEG DIS)

9.2.4 Operations

In the MHEG API specification, primitives should be defined as follows:

- request primitives should be specified as either operations provided by object interfaces or accessors and modifiers of the attributes provided by object interfaces. Requests will be either synchronous (client suspended until response is provided), deferred synchronous (response provided, client not suspended) or asynchronous (no response);

- response primitives should be specified as the results of the synchronous operations describing requests;

- notification primitives should be specified as either exceptions that may be raised by requests or as events. The latter category applies mainly for the return actions, which are part of the behaviour function category. A syntax for the definition of events (currently not supported by IDL) should be provided by item 3) as mentioned in subclause 9.1.

The following is an indicative list of the operations and attributes that the interfaces of the objects defined in subclause 8.2.1 should provide as the MHEG API. This informal list should be completed as necessary and used as the basis for the formal IDL specification.

MHEGEngine

- OpenSession
- CloseSession
- GetMHEGEngineCapabilities

MHEGObjectManager

- SelectObjects (according to attribute, location and/or other criteria)
- SupplyObject (make it available to the application)
- DeleteObject

MHEGObject

- attribute: logical name(s)
- MHEG attributes: standard version, class identifier, MHEG identifier, name, owner, version, date, keywords, copyright, licence, comments
- Send
- Retrieve
- Store

MHEGAction

- MHEG attributes: synchro indicator, performances, target set, synchronised actions (attributes should be organized so as to allow accessing, modifying, adding or removing one identified elementary action or nested action object within the object)

MHEGLink

- MHEG attributes

MHEGContainer

- MHEG attributes

MHEGDescriptor

- MHEG attributes

MHEGModel

- MHEG attributes

MHEGScript

- MHEG attributes

MHEGComponent

- MHEG attributes

MHEGComposite

- MHEG attributes

MHEGContent

- MHEG attributes

MHEGMultiplexedContent

- MHEG attributes

EntityManager

- SelectEntities (according to attribute, status or other criteria)

Entity

- attribute: alias(es)

mhObject

- readonly attribute: original MHEG object
- Prepare: takes an MHEG object as argument, sets the original MHEGobject attribute
- Destroy
- readonly attribute: preparation status (modified by Prepare and Destroy operations)

NOTE 1: As recommended by both MHEG and IDL, operations that modify the life status of objects, i.e. creation and destruction operations, are provided by the created (or destroyed) object. The original object on which to base for creation (the reference to an MHEGObject for the creation of an mhObject, the reference to an mhModel for the creation of an rtObject) is provided as an argument.

rtObject

- readonly attribute: original mhModel
- New: takes an mhModel as argument, sets the original mhModel attribute
- Delete
- readonly attribute: availability status, running status
- Run
- Stop

NOTE 2: List of object operations to be completed according to behaviour table in subclause 9.2.3.

10 SIR functional requirements

10.1 Synthesis of application requirements

The requirements for scripts, for the services and applications examined in clause 6, are given in table 29.

Table 29: Requirements for scripts

Scripts required for each application	Encyclopeadia	PoI PoS	Telematic training	EPG
External device control	✓	✓	✓	✓
External device control for data acquisition				✓
Manipulation of MHEG objects	✓			✓
Access to external data	✓	✓		✓
Access to external advanced calculation capability		✓		
Computations, variable handling and control structures	✓	✓	✓	✓
Key: PoI: Point of Information PoS: Point of Sale				

Scripts required for each application	Teleshopping	iVoD	Video-telephony Games	Tele-control
External device control	✓			✓
External device control for data acquisition	✓	✓		✓
Manipulation of MHEG objects	✓	✓	✓	✓
Access to external data	✓	✓		
Access to external advanced calculation capability			✓	✓
Computations, variable handling and control structures	✓		✓	

The overall requirements for the applications above, are as follows:

- external device control;
- external device control for data acquisition;
- manipulation of MHEG objects;
- access to external data;
- access to external advanced calculation capability;
- computations, variable handling and control structures.

The way in which the scripts are implemented needs to be determined.

10.1.1 Terminology

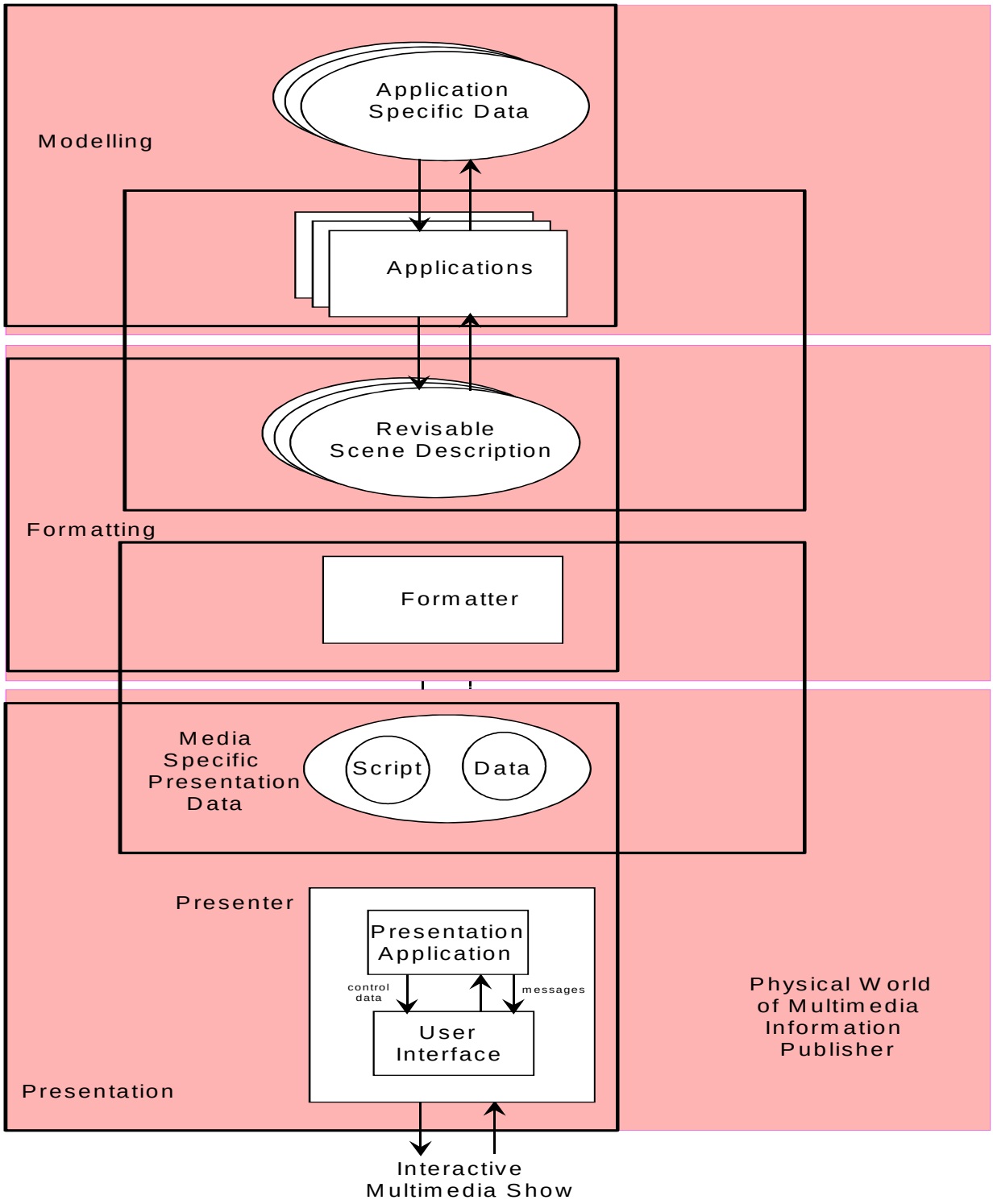

Figure 13: Production, interchange and use of multimedia

The above figure, originally developed to be a reference model for multimedia, is used in this ETR to illustrate the process of production, interchange and use of scripts in multimedia applications.

The figure shows a number of information repositories (pertaining to information modelling, formatting and presentation processes) linked together by common objects or procedural APIs. The figure includes the following:

- **Application Specific Data** which is used primarily to interchange information between applications co-operating in the process of information creation;

- **Revisable Scene Description** where application specific semantically meaningful objects are already associated with information visualisation and presentation specific attributes;

- **Media Specific Presentation Data** where specified associations between semantics of particular user application, and visualisation and presentation attributes are already resolved to the form of a content suitable for multimedia presentation of original information.

Repositories containing revisable scene description, or media specific presentation data, allow through common and often standardized semantics, to distribute, share, store and retrieve multimedia information without direct access to an application originating this information. In such context, if information user and information provider are spatially or temporary separated, two additional processes are introduced, namely:

a) **Scene Creation Process** which provides for an application independent platform to integrate information in the process of visualisation, and for common input platform to a formatter;

b) **Information Publishing Process** providing for an important and simple way to distribute final form presentation either:

   1) in open public interchange format like MHEG and MHEG-S; or

   2) in proprietary format often dedicated to some specific presentation hardware and software.

For precise evaluation of direct environment around MHEG standard, the presentation part of the basic reference model is split into **presentation application** and **user interface process**. In the process the information verification, they will exchange messages and control information visualising data already available in some standard or proprietary procedural (script) or descriptive forms.

More than one presentation application can share the same media specific presentation data using one or more user interface processes. They then constitute the **presentation environment**.

10.1.2 Scripting Languages, Scripts and Script Interchange Representation

A Scripting Language (SL) is a programming language, however not all programming languages are SLs. More precisely, the set of SLs is a subset of programming languages. The attributes which differentiate them are subtle, and often have to do with marketing as well as with technical distinctions. A SL is used in the Scene Creation Process (see figure 13).

SLs usually display one or more of the following characteristics:

- language syntax and grammar resemble natural (spoken) languages;

- language is usable by people who are not professional programmers;

- language design stresses simplicity over functionality;

- language is application focused, in that it is intended to address the need of one category of application (language is not "general purpose").

The programs written in SL are called "**Scripts**".

The **SIR** is a coded representation used by an application to interchange scripts in the purpose of their processing on a remote device. From one script one or several (interpretable or not, executable or not, reprocessable or not, re-editable or not) may usually be generated. The main advantage a SL offers compared to the SIR is that to express very easily the desired functionality. A SIR is used in the Information Publishing Process (see figure 13).

The high level requirements for SLs the applications presented in clause 6 have are very different. It is therefore very difficult to fulfil them with only one set of functionality. If only the SIR is standardized than application providers can use a specific SL that is best suited to their application to design scripts and translate them afterwards into the standardized SIR.

The SIR should therefore be specified hardware independent using only a basic set of low-level commands that in their combination are suitable to provide all desired functionality.

10.2 Functional scope of SIR

The overall requirements for the applications described in clause 6 of this ETR, are as follows:

- external device control;
- external device control for data acquisition;
- manipulation of MHEG objects;
- access to external data;
- access to external advanced calculation capability;
- computations, variable handling and control structures.

In order to support the application requirements, the SIR chosen should provide the following:

a) the ability of MHEG objects to interface with external processes in a standardized way;
b) the ability to deal with minimal resource constraints.

The functions derived from the application requirements can be split into four areas. These functions should be expressible by the SIR. The four function areas are as follows:

1) handling of data (variables, collections);
2) operations on data (computing, comparison, assignment, etc,...);
3) co-operation with MHEG objects (sending and receiving actions);
4) co-operation with external processes (activation and deactivation, sending and receiving data, etc,...).

11 Technical requirements on the MHEG SIR

This clause addresses the technical aspects of the MHEG SIR standard specification.

For this purpose, this clause:

- states the objective and scope of the MHEG SIR, from a technical perspective;
- recommends technical options;
- recommends a technical framework for the drafting of the MHEG SIR standard.

11.1 Technical requirements on the MHEG SIR specification

The technical requirements for the target representation are the following:

- hardware independence;
- final form;
- mechanisms for interfacing with external processes or libraries such as floating point arithmetic packages (for example to calculate simulations, etc,...);
- compactness (small size of code);
- easy to implement simple and powerful interpreters (reduced instructions set);
- open and extensible structures;
- allowing real-time interchange;
- allowing to be produced by specialised tools;
- resistant to reverse engineering;
- checkability of the syntax of the interchanged data for quality of service purposes;
- checkability of the interchanged data for contamination purposes for security and protection purposes;
- a non-proprietary representation allowing developers and implementors to be free from copyright issues.

11.2 Technical options

11.2.1 Architecture Neutral Distribution Format (ANDF)

One candidate for a SIR is the Architecture Neutral Distribution Format (ANDF) which was developed by the TDF team at the Defence Research Agency (DRA) in Malvern, UK. This development meets the requirements of architectural neutrality and is resistant to reverse engineering. An ANDF allows software developers to develop software as usual in high level languages (C and FORTRAN, etc,...) and then compile it into a distribution format. In this ANDF form, it is then distributed to users, who use local compilers specific to their own architectures to compile it to the appropriate object code. DRA's TDF technology amounts to a new compiler language. ANDF is not primarily intended for real-time interchange.

11.2.2 p-code

Most compilers translate scripts into machine code that a computer can execute directly. With special compilers programs can be compiled into an alternate format called p-code. p-code produces much smaller programs than machine code but a computer cannot execute them directly. Instead, programs compiled into p-code are executed by an interpreter. As a result p-code programs are slower than machine code programs.

p-code is object code consisting of instructions for a pseudo-machine - an idealised computer architecture optimised for high-level language execution on small host machines. Object p-code may originate from any one of a number of source languages (Pascal, FORTRAN, Basic, etc,...). As long as the program has no native code elements, it is isolated entirely from the host computer and can be executed without recompilation and indeed without modifications of any kind on most, if not all, host hardware.

When native code is required, there are two ways to generate it. First, the native code can be produced automatically by a code-generator program that takes the p-code as input and produces equivalent native code for programmer-designated performance-critical sections. No matter how the native code is produced, its execution can be interleaved with p-code execution. That is, a program can have both native and p-code components, and frequent transitions back and forth between the two modes of execution.

11.2.3 Other options - intermediate languages

Kaleida is a joint venture of IBM and Apple. The major product of this company is a scripting language called Script/X, which is based on object-oriented technology and the C programming language. Script/X ventures away from the usual scripting language in that it provides for an intermediate language that is independent of the source language. In theory, script can be developed in any language and then compiled into Kaleida's proprietary intermediate form. This approach has been applied to programming languages before: p-code and ANDF are examples of intermediate forms that are currently used by commercial compilers.

With Script/X, which Kaleida describes as a device independent, object-oriented multimedia description language, developers can write to a uniform set of APIs, independent from any hardware. To make the applications play on many different machines, Kaleida will create what it calls run-time engines (RTEs) for each platform. The basis of the company's business is the proprietary-intermediate format and machine-specific RTEs which Kaleida will license to both hardware manufactures and software developers.

In the hope that authoring tool developers will replace with Script/X their proprietary scripting language for performance and interchange purposes, and that Script/X will be the industry language of choice for specification of procedures which might be referenced from HyTime, Kaleida submits the Script/X specification to Interactive Multimedia Association (IMA) in response to that group's multimedia scripting language **Request for Technology** despite the fact that the work on the scripting language is not complete yet.

11.2.4 Conclusion

For the Kaleida intermediate forms, work is not complete and they are proprietary in nature.

Given the advantages of p-code, and the fact that ANDF is not primarily intended for real-time interchange, and furthermore, does not meet the functional and performance requirements, p-code is the recommended SIR out of the presented options.

11.3 Guidelines for drafting the MHEG SIR standard

To define a SIR that lies on an optimised level between a high level scripting language or programming language (C, Pascal, etc,...) and a low level assembly language is a very difficult task of this project. The following guidelines should be taken into account when making the final decision on which level to define the SIR.

The definition of a SIR too close to the level of a programming language would cause the following problems:

- the code would not be resistant to reverse engineering;

- the SIR would not be final form so it would not be interpretable in real-time. It would be more difficult to build SIR interpreters;

- it will not be possible to produce compilers for all different scripting languages that can compile the scripts into SIR.

The definition of a SIR too close to the level of a the assembly language would cause the following problems:

a) the SIR would not be as compact as with a high level language;
b) the SIR would not be machine independent.

A recommended approach to develop an optimised SIR is to start on the basis of a near assembly code language and increase the functionality until all of the requirements for a SIR are fulfilled.

11.4 Possible implementation example using p-code

The following subclauses are based on the assumption that the SIR implementation will use a p-code like concept. However, most of the described functionality will be similar even if a different solution than a p-code like stack machine is used.

11.4.1 The p-code stack machine

While machine language consists of instructions for the microprocessor in a computer, p-code consists of instructions for an imaginary processor that is simulated by an run-time interpreter. This imaginary processor is known as "stack machine", because it uses a stack for almost all of its operation. In contrast, the microprocessor in a computer uses its registers for most of the operations and uses the stack preliminary for function calls.

The stack holds the operands used by the instructions. In assembly language usually a source and a destination is defined for any instruction, indicating where the operands reside and where to place any results (see example 1).

EXAMPLE 1: ADD AX, BX

With a stack machine usually no source and destination is defined. Each instruction pops its operands off the stack and pushes its results back onto the stack (see example 2).

EXAMPLE 2: The p-code instruction

```
ADDW
```

would imply the following operations (using C-like pseudo code):

```
w2    =     pop()  // get first operand from the stack
w1    =     pop()  // get second operand from the stack
push(w1+w2)      // place result on the stack.
```

Omitting the source and destination saves space and helps to make p-code as compact as it is. This is the main advantage of using a stack machine based pseudo-architecture instead of a register machine based pseudo-architecture.

The stack can store items of different data types, including integer, real etc,... It replaces the need for the general purpose registers (AX through DX).

11.4.2 Relationship between script engine and MHEG engine

The information exchange between the script engine and the MHEG engine is handled via the MHEG API. The Script engine is the client of the MHEG engine and as such can use all the API services offered. Using this MHEG API primitives a script instance can interact on all available MHEG objects.

With the starting of the execution of each script the script engine has to start a new instance of the script interpreter. Every instance of this script interpreter should run one instance of a script. The script instances can exchange information with each other and can communicate with the outside world. Synchronisation between the script instances may be needed in several specific cases. The synchronisation could be provided by messages between the script instances and a kind of "Wait for message" instruction that would suspend the execution of a script instance until a specific message is received.

11.4.3 The p-code data types

The following aspects should be taken into account when selecting the data types to be used in the p-code:

- platform independence should be guaranteed;

- a minimal number of data types to support all functionality should be provided. Care should be taken not to make the p-code too complex by introducing too many different data types;

- basic data types that allow the using application to process any kind of data by bringing application specific semantics to the used structures should be implemented;

- the expression of more complex data types like records or arrays should be enabled either by providing access facilities to different offsets within one bytestring, or by defining more complex ASN.1 like structures;

- harmonization with the data structures of the notation that is used to encode the SIR should be envisaged.

Taking into account all previously made considerations, the following datatypes are proposed (the list of data types may be amended or changed according to requirements arising from the actual implementation of the SIR in later phase of work:

1) integer;
2) real;
3) byte string;
4) character;
5) enumerated;
6) object identifier;
7) object descriptor;

8) complex data types built from constructors (such as, sequence, set, choice, sequence of, set of, etc,...).

11.4.4 Extension mechanisms using a Call instruction

The Call instruction enables the invocation of functions or external processes allowing the exchange of parameters in both ways. The following two scenarios can be identified:

- direct call;
- indirect call.

The direct Calls invokes a function different from the currently executing function. An indirect Call invokes either a function unknown at design time, a function in a prototype package (library of often used SIR function to ease development) or any kind of external process. Both Call instructions should support the exchange of parameters in both directions.

This Call instruction provides a powerful means to interface with external processes and to enlarge the capabilities of the SIR in order to fulfil the requirement of all different families of services.

11.4.5 SIR instructions

The following subclauses give an overview of the instruction families needed to implement the SIR. The description of the instructions that might be useful is not exhaustive but can be used as a basis for standardising the SIR. During the development of a SIR ETS additional instructions are likely to be added and others are likely to be suppressed.

11.4.6 SIR Header

A header carrying the identification of the script should be defined.

11.4.7 Declaration of variables

The following two categories of variables could be useful:

- global variables;
- local variables.

The lifetime of global variables is the whole activation time of the current instance. The lifetime of local variables is limited to the execution time of the current subroutine. There should be instructions to declare global and local variables for each defined data type.

An instruction for the declaration of constants for the basic data types may be available as well.

11.4.8 Stack management

For the stack management Push and Pop instructions should be defined. The number of bytes to pushed or popped is determined by the variable to which the instruction applies. Instructions to Push or Pop subparts of byte string are needed as well.

Instructions to clear items from the stack (bytewise or itemwise) should be available as well.

11.4.9 Conversions

Conversion instructions between the different basic data types should be defined.

11.4.10 Byte manipulation and logical operators

Following instructions are applicable only to integer value:

- increment;
- decrement.

Following instructions are applicable to integer and real:

- add;
- subtract;
- multiply;
- divide.

Following logical operators should be available:

- and;
- or;
- exclusive or;
- not;
- complement to one.

Comparison instruction should be available to compare two items on the stack. A possible implementation of the compare could be to Pop two elements from the stack and perform a compare on them. Depending on the results of the compare, defined value is pushed onto the stack.

11.4.11 Control structures

An unconditional jump instruction should be available. The following two possible options are available for implementing the unconditional jump:

1) the target of the jump is defined by a label;
2) the displacement in number of items is given in the jump instruction itself.

To reduce the code size of the SIR it should be considered to implement various branch instructions. Due to the fact that branch and case operations will occur very often in the target families of applications a more extensive instruction set for branch operations might enable the SIR compilers to reduce the code size significantly. The following branch operations are suggested:

- Branch on Equal;
- Branch on Greater or Equal;
- Branch on Greater Than;
- Branch on Less than or Equal;
- Branch on Less Than;
- Branch on Not Equal;
- Branch on Not Zero;
- Branch on Zero;
- Case.

The Case instruction executes one of several statement blocks depending on the value of an expression.

11.4.12 Subroutine definition

The definition of subroutines within the SIR is especially valuable to reduce program code. A subroutine should be preceded by a subfunction header and terminated by a return statement.

11.4.13 Additional instructions

An instruction to immediately terminate the execution of the script instance is needed. Also the suspension of the execution of a script instance should be possible. In both cases the control should be given back to the calling entity and a parameter exchange between the script instance and the calling entity should be enabled.

To enable the information exchange and the synchronisation between different scripts a messaging mechanisms is suggested. For synchronisation purposes a "Wait for message" instruction and a "Send Message" should be defined.

An instruction that suspends the execution of a script instance for a given time without giving the control back to the calling entity should be available as well (Wait).

11.5 Aspects relating to the notation for interchange

A SIR will allow the interchange of instructions as well as some mechanisms for helping with the interpretation. Two notation methods are possible candidates - ASN.1 and SGML.

Given that SGML is not intended for real-time interchange, and ASN.1 is already used for the coded representation of MHEG it is proposed to describe the SIR using ASN.1 as a base notation.

History

Document history	
October 1995	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)

ISBN 2-7437-0286-9
Dépôt légal : Octobre 1995