INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.174

(10/96)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Application Programming Interface (API) for MHEG-1

ITU-T Recommendation T.174

(Previously CCITT Recommendation)

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

For further details, please refer to ITU-T List of Recommendations.

FOREWORD

ITU-T Recommendation T.174 was prepared by ITU-T Study Group 8 (1993-1996) and was approved by the WTSC (Geneva, 9-18 October 1996).

____

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

CONTENTS

Page

SUMMARY

This Recommendation specifies the abstract Application Programming Interface (API) for the manipulation of multimedia and hypermedia information objects, i.e. the API that shall be provided by MHEG engines for their control by MHEG applications.

INTRODUCTION

This Recommendation specifies the abstract Application Programming Interface (API) for the manipulation of multimedia and hypermedia information objects, i.e. the API that shall be provided by MHEG engines for their control by MHEG applications.

This Recommendation is part of a broader standardisation framework that specifies the usage of MHEG so that interoperable equipments can be effectively developed to support multimedia information services and applications. This implies:

– specifying additional constraints to the use of MHEG objects within distributed systems and applications using telecommunication networks;

– defining APIs that building blocks of MHEG-using architectures should provide;

– defining MHEG profiles complementing the MHEG-1 standard by specifying restrictions on the coded representation and specifying the complete required behaviour of an MHEG engine that should be supported for a given category of applications and/or terminal equipments;

– defining an MHEG script interchange representation;

– defining end-to-end protocols for MHEG-using multimedia/hypermedia information services;

– specifying conformance testing procedures for these standards.

Functional and technical requirements on this Recommendation have been described in ETR 225.

Recommendation T.174

APPLICATION PROGRAMMING INTERFACE (API) FOR MHEG-1

(Geneva, 1996)

1 Scope

MHEG Part 1 (ISO/IEC 13522-1 [1]) is a generic standard, which specifies the coded representation of interchanged multimedia/hypermedia information objects (MHEG objects). These so-called MHEG objects are handled, interpreted and presented by MHEG engines.

This Recommendation specifies the abstract Application Programming Interface (API) for the manipulation of multimedia and hypermedia information objects, i.e. the API that shall be provided by MHEG engines for their control by MHEG applications.

This API meets the following requirements:

- it is independent of the programming language used for the MHEG application;
- it is independent of the underlying operating system;
- it is independent of the mechanism used for interchanging information between the API user (i.e. MHEG application) and the API provider (i.e. MHEG engine), i.e. the messages that are exchanged as the result of triggering API primitives;
- it is independent of the actual encoding of these messages;
- it is generic and meant to cover all application requirements;
- it is conformance testable;
- it aims to be as easy as possible to implement.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[1] ISO/IEC 13522-1:1997, *Information technology – Coding of multimedia and hypermedia information – Part 1: MHEG object representation – Base Notation (ASN.1).*

[2] ITU-T Recommendation X.290 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – General concepts.*

[3] ETR 173: *Terminal Equipment (TE) – Functional Model for Multimedia Applications*.

[4] ETR 225: *Terminal Equipment (TE) – API and script representation for MHEG – Requirements and framework.*

[5] CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1)*.

[6] CCITT Recommendation X.209 (1988), *Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)*.

[7] ITU-T Recommendation I.113 (1993), *Vocabulary of terms for broadband aspects of ISDN*.

[8] ITU-T Recommendation I.112 (1993), *Vocabulary of terms for ISDNs*.

[9] CCITT Recommendation Q.9 (1988), *Vocabulary of switching and signalling terms*.

[10] ISO/IEC 14750[1], *Information technology – Open Distributed Processing – Interface Definition Language*.

[11] ISO/IEC JTC 1 N 2965:1994, *Guidelines for JTC 1 API Standardisation*.

3 Definitions and abbreviations

3.1 Definitions

Due to the particular nature of this Recommendation, some of the words and expressions used in this Recommendation come from the ’telecommunication services’ standards glossary, while others come from the ’software technology’ standards glossary. This leads to words whose meaning vary according to the context, i.e. the expression within which they are used. For this reason, many of these expressions are defined in this subclause.

Should any ambiguity occur, definitions of the following Standards/Recommendations would apply, in decreasing order:

- ISO/IEC 13522-1 [1] MHEG;
- any other standard part of ISO/IEC 13522 [1] MHEG;
- CCITT Recommendation I.113 [7], "Vocabulary of terms for broadband aspects";
- CCITT Recommendation I.112 [8], "Vocabulary of terms for ISDNs";
- CCITT Recommendation Q.9 [9], "Vocabulary of switching and signalling terms".

This Recommendation defines the following terms:

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

3.1.2 function family: Cluster of functional MHEG-API requirements consisting of functions with related semantics and applying on a same type of target.

3.1.3 hypermedia: The ability to access monomedia and multimedia information by interaction with explicit links.

3.1.4 interactive service: A service which provides the means for bidirectional 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 (CCITT Recommendation I.113 [7]).

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

3.1.6 MHEG-API: The API provided by an MHEG engine to MHEG applications for the manipulation of MHEG objects, as defined in this Recommendation.

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

3.1.8 MHEG engine: A process or a set of processes that interpret MHEG objects encoded according to the encoding specifications of ISO/IEC 13522-1 [1]: Abstract Syntax Notation One (ASN.1) for Part 1, Standard Generalized Markup Language (SGML) for Part 2.

3.1.9 MHEG using application: An application which involves the interchange of MHEG objects within itself or with another application.

[1] To be published.

3.1.10 **multimedia and hypermedia application**: An application which involves the presentation of multimedia information to the user and the interactive navigation across this information by the user.

3.1.11 **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.

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

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

3.1.14 **primitive**: One of the basic entry points provided by a provider module to any user module to enable the user module to access the software service(s) supplied by the provider module.

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

3.1.16 **(software) service**: A set of functions provided by a (server) software or system to a client software or system, usually accessible through an application programming interface.

3.1.17 **(telecommunication) application**: A set of a user's requirements (see Recommendation Q.9 [9]).

3.1.18 **(telecommunication) service**: That which is offered by an Administration to its customers in order to satisfy a specific telecommunication requirement (see Recommendation I.112 [8]).

3.1.19 **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.

3.1.20 **user**: A person or machine delegated by a customer to use the services and/or facilities of a telecommunication network (see Recommendation I.112 [8]).

3.1.21 **action (object)**: An object that provides operation on objects, e.g. to change their attributes or states.

3.1.22 **channel**: A logical space in which rt-components are positioned for final presentation. Channels are mapped by the MHEG engine to physical devices like screen windows or loudspeaker for making the rt-objects within them perceivable by the user.

3.1.23 **component (object)**: An abstraction which represents objects of Content or Composite type.

3.1.24 **composite (object)**: A list of Composition Elements grouped for presentation. The presentation of a Composite object consists of the presentation of its Composition Elements.

3.1.25 **container (object)**: A means to group objects without specifying specific relationships.

3.1.26 **content (object)**: Encoded generic value, media or non-media data.

3.1.27 **descriptor (object)**: A structure for the interchange of resource information about a single or a set of other interchanged objects.

3.1.28 **(IDL) attribute**: An identifiable association between an object and a value. An attribute **A** is made visible to clients as a pair of operations: **get_A** and **set_A**. Read only attributes only generate a **get** operation.

3.1.29 **(IDL) class**: An implementation that can be instantiated to create multiple objects with the same behaviour. An object is an instance of a class. Types classify objects according to a common implementation.

3.1.30 **(IDL) data type**: A categorisation of values operation arguments, typically covering both behaviour and representation (i.e. the traditional non-Object Oriented (OO) programming language notion of type).

3.1.31 (IDL) instance: An object is an instance of an interface if it provides the operations, signatures and semantics specified by that interface. An object is an instance of an implementation if its behaviour is provided by that implementation.

3.1.32 (IDL) object: A combination of state and a set of methods that explicitly embodies an abstraction characterised by the behaviour of the relevant requests. An object is an instance of an implementation and an interface. An object models a real-world-entity, and it is implemented as a computational entity that encapsulates state and operations (internally implemented as data and methods) and responds to requester services.

3.1.33 (IDL) operation: A service that can be requested. An operation has an associated signature, which may restrict which actual parameters are valid.

3.1.34 interface definition language (IDL): A language that specifies types and objects by specifying their interface. It provides a conceptual framework for describing the objects.

3.1.35 link (object): An object which defines spatio-temporal relationships between other objects.

3.1.36 macro (object): An object that provides the facility to replace the parameters in frequently used action objects.

3.1.37 Mh-object: MhObjects (and their subtypes) match form b) objects as defined in ISO/IEC 13522-1: "Information technology – Coding of multimedia and hypermedia information" [1], 6.2.4, i.e. objects available to the MHEG engine.

3.1.38 (MHEG) elementary action: An attribute of the MHEG Action class which instructs an object to perform a certain operation, e.g. to change one of its attributes or states.

3.1.39 MHEG entity: Any MHEG object, rt-object, content data, script data, socket, channel or other construction identified or referred to in the MHEG standard.

3.1.40 MHEG interpretation service: The service takes interchanged MHEG objects and messages issued by the presentation system as an input. It analyses this data in order to trigger the presentation of content data according to its semantics.

3.1.41 MHEG object handling service: This facility physically creates, handles and maintains intermediate data structures necessary to implement a client access to the MHEG objects and their contents.

3.1.42 MHEG object: A coded representation of an instance of an MHEG object class.

3.1.43 model object: Object instantiations from the MHEG classes.

3.1.44 notification: A primitive issued by the server on its own initiative to forward information to the client.

3.1.45 request: A primitive issued by the client on its own initiative to forward information to the server.

3.1.46 response: A primitive issued by the server as a reply to a request to forward information to the client.

3.1.47 rt-object: RtObjects (and their subtypes) match form c) objects as defined in ISO/IEC 13522-1: "Information technology – Coding of multimedia and hypermedia information" [1], 6.2.4, i.e. instances of MhObjects available to the presentation process.

3.1.48 script (object): Object that provides the structure to interchange script data in a specified encoded form.

3.1.49 socket: An element of an rt-composite. Rt-components are plugged into sockets. Different types of sockets are defined depending on the rt-component plugged into the socket:

– empty socket, i.e. a null rt-component is plugged;

– presentable socket, i.e. an rt-content or an rt-multiplexed content is plugged;

– structural socket, i.e. an rt-composite is plugged.

3.2 Abbreviations

This Recommendation uses the following abbreviations:

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One (as defined in ITU-T Recommendation X.208 [5])
BER	Basic Encoding Rules (as defined in Recommendation X.209 [6])
C1	State period for a Channel: ’non-available’
C2	State period for a Channel: ’processing’
C3	State period for a Channel: ’available’
C4	State period for a Channel: ’processing’
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CGS	Channel Generic Space
CGSU	Channel Generic Space Unit
CORBA	Common Object Request Broker Architecture
DIS	Draft International Standard
EBNF	Extended Backus-Naur Form
ETR	ETSI Technical Report
ETS	European Telecommunication Standard
ETSI	European Telecommunications Standards Institute
GSR	Generic Spatial Ratio
GTU	Generic Temporal Unit
IDL	Interface Definition Language (as defined in ISO/IEC 14750-1 IDL [10])
IEC	International Electrotechnical Commission
IOGTR	Initial Original generic space Generic Temporal Ratio
IOV	Initial Original audible Volume
ISO	International Organisation for Standardisation
ITU-T	International Telecommunication Union – Telecommunication Standardisation Sector
JTC	Joint Technical Committee
MCU	Multipoint Control Unit
MHEG	Multimedia and Hypermedia information coding Experts Group
MPEG	Moving Picture Experts Group
O1	State period for an mh-object: 'not ready'
O2	State period for an mh-object: 'processing'
O3	State period for an mh-object: 'ready'
O4	State period for an mh-object: 'processing'
OGS	Original Generic Space
OGSU	Original generic space Generic Spatial Unit
OGTR	Original generic space Generic Temporal Ratio

OGTU	Original generic space Generic Temporal Unit
OO	Object Oriented
OSI	Open Systems Interconnection
OV	Original Volume
PRGS	Parent Relative Generic Space
PS	Perceptible Size
PSAP	Perceptible Size Attachment Point
R1	State period for an rt-object: ’not available’
R2	State period for an rt-object: ’processing’
R3	State period for an rt-object: ’available’
R4	State period for an rt-object: ’processing’
RGS	Relative Generic Space
RGSU	Relative generic space Generic Spatial Unit
RGTU	Relative generic space Generic Temporal Unit
rt	Run-time
SGML	Standard Generalized Markup Language
SIR	Script Interchange Representation
SSU	Service Support Unit
VD	Visible Duration
VS	Visible Size
VSEAP	Visible Size External Attachment Point
VSGS	Visible Size Generic Space
VSGSU	Visible Size Generic Spatial Unit
VSIAP	Visible Size Internal Attachment Point

4 Conformance

An implementation of this Recommendation is an MHEG engine implementation which provides client MHEG applications with one or several language bindings of the abstract Application Programming Interface (API) defined in this Recommendation.

An application of this Recommendation is an MHEG application which uses a language binding of the abstract API defined in this Recommendation to control the behaviour of an MHEG engine.

The following subclauses state requirements associated with the conformance of both implementations and applications to this Recommendation.

4.1 Implementation conformance

4.1.1 Conformance requirements

A conforming implementation for a language binding specification for this Recommendation shall meet all of the following criteria:

1) The implementation shall support all required behaviour defined in this Recommendation.

2) The implementation shall support all required interfaces defined in the language binding specification. Those interfaces shall support the behaviour described in this Recommendation and in the language binding specification.

3) The implementation may provide additional functions or facilities not required by this Recommendation or by the language binding specification. Each such non-standard extension shall be identified as such in the system documentation. Non-standard extensions, when used, may change the behaviour of functions or facilities defined by this Recommendation or by the language binding specification. The conformance document shall define an environment in which an application can be run with the behaviour specified by this Recommendation and the language binding specification. In no case shall such an environment require modification of a Strictly Conforming Application.

4.1.2 Conformance documentation

A conformance document with the following information shall be available for an implementation claiming conformance to a language binding specification for this Recommendation. The conformance document shall be in two parts. The first part shall have the same structure as this Recommendation, with the information presented in the appropriately numbered sections, clauses, and subclauses. The second part shall have the same structure as the language binding specification, with the information presented in the appropriately numbered sections, clauses, and subclauses. The conformance document shall not contain information about extended features or capabilities outside the scope of this Recommendation and the language binding specification.

The conformance document shall identify the language binding specification to which the implementation conforms.

The conformance document shall contain a statement that indicates the full names, numbers, and dates of the language-independent and language binding specification standards that apply.

The conformance document shall state which of the optional features defined in this Recommendation and in the language binding specification are supported by the implementation.

The conformance document shall describe the behaviour of the implementation for all implementation-defined features defined in this Recommendation and in the language binding specification. This requirement shall be met by listing these features and by providing either a specific reference to the system documentation or full syntax and semantics of these features. The conformance document may specify the behaviour of the implementation for those features where this Recommendation or the language binding specification states that implementations may vary or where features are identified as undefined or unspecified.

No specifications other than those specified by this Recommendation and the language binding specification shall be present in the conformance document.

The phrases "shall document" or "shall be documented" in this Recommendation or in a language binding specification for this Recommendation mean that documentation of the feature shall appear in the conformance document, as described previously, unless the system documentation is explicitly mentioned.

The system documentation should also contain the information found in the conformance document.

4.2 Application conformance

All applications claiming conformance to a language binding specification for this Recommendation shall fall within one of the categories defined in the following subclauses.

4.2.1 Strictly Conforming Application

A Strictly Conforming Application is an application that requires only the mandatory facilities described in this Recommendation, in the language binding specification and in the applicable language standards. Such an application shall accept a behaviour described in this Recommendation or in the language binding specification as unspecified or implementation defined and, for symbolic constants shall accept any value in the ranges permitted by this Recommendation and the language binding specification.

4.2.2 Conforming Application

A Conforming Application of a language binding specification for this Recommendation is an application that differs from a Strictly Conforming Application in that it may use optional facilities described in this Recommendation, in the language binding specification and in the applicable language standards, as well as non-standard facilities that are consistent with the standard and with the language binding specification. Such an application shall fully document its requirements for these optional and extended facilities in addition to the documentation required of a Conforming Application.

4.3 Test Methods

Any measurement of conformance to a language binding specification for this Recommendation shall be performed using test methods that conform to Recommendation X.290 [2] and to any additional requirements that may be imposed by the language binding specification.

5 General description

This clause situates the MHEG-API within the architecture of an MHEG using application. It then specifies and classifies the software services that the MHEG API shall provide to its client applications.

5.1 Functional reference model of MHEG using applications

5.1.1 Reference model for multimedia applications

ETR 173 [3] "Functional Model for Multimedia Applications" and ETR 225 [4] "API and script representation for MHEG – Requirements and framework" define a generic reference model describing a functional architecture common to all multimedia applications making use of retrieval, conversational and/or distribution services. The model is applicable to all MHEG using applications.

Figures 1 to 3 show how this reference model applies to MHEG using applications for terminal-to-host, terminal-to-terminal and terminal-to-database configurations in both point-to-point and multipoint communication.

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

1) presentation agent;

2) access agent;

3) local application interpreter;

4) MHEG engine;

5) local information base.

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

1) the services of the presentation agent are offered through the presentation API;

2) the services of the MHEG engine are offered through the MHEG-API;

3) the services of the access agent are offered through the access API.

In an 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.

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.

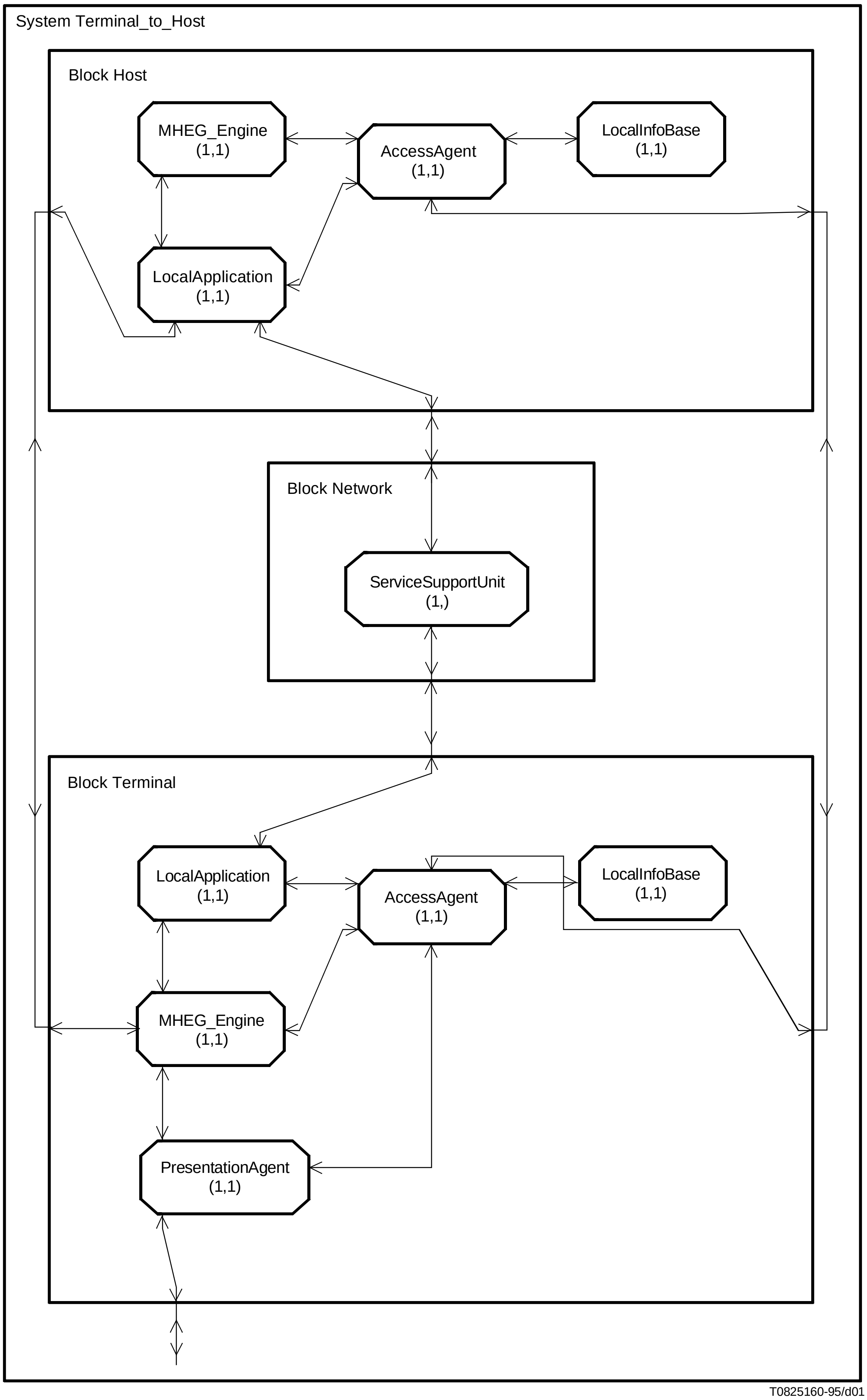


Figure 1/T.174 – Application architecture reference model for terminal-to-host configurations

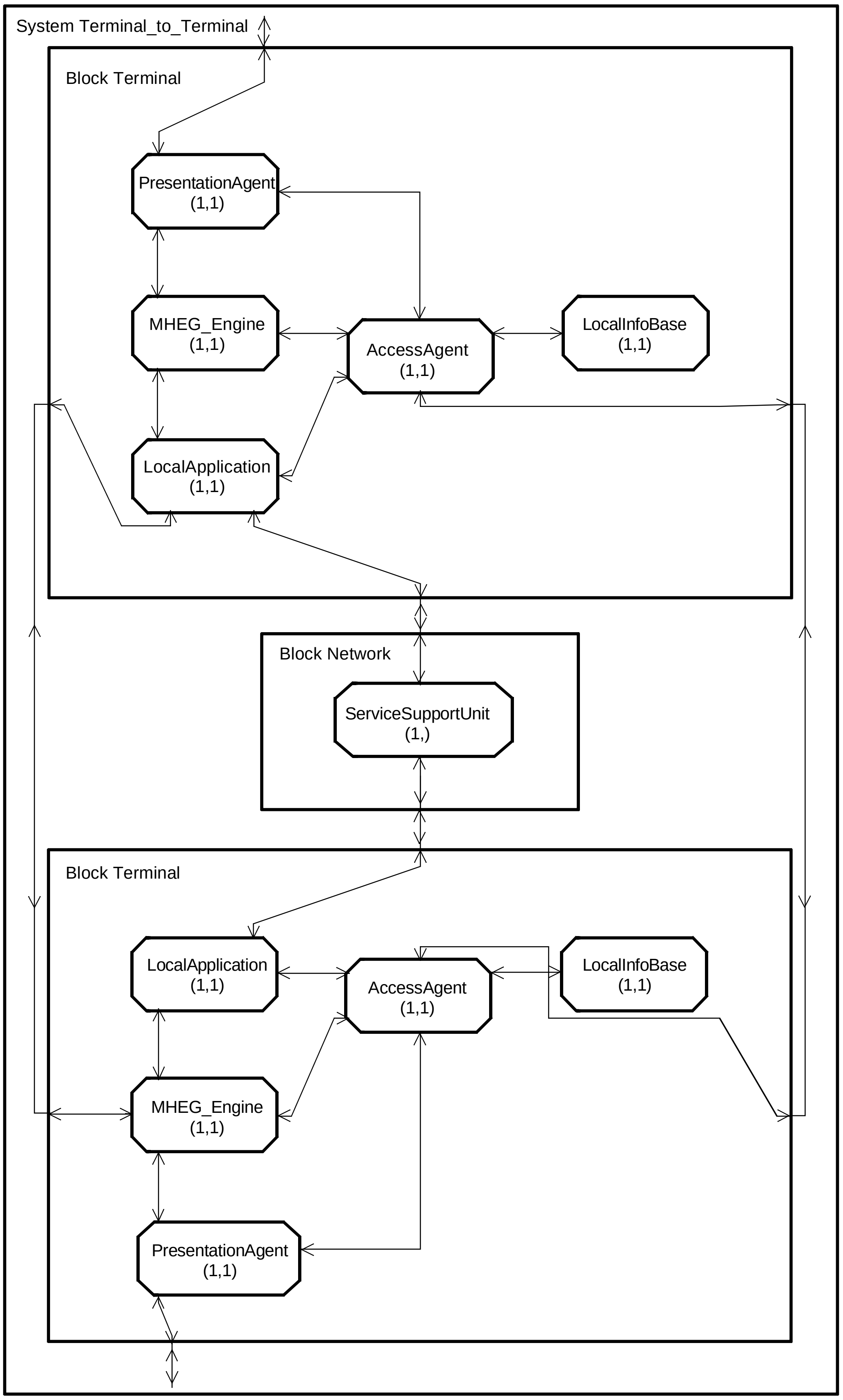


Figure 2/T.174 – Application architecture reference model for terminal-to-terminal configurations

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 **presentation agent** provides a multimedia content presentation service. It manages the presentation of monomedia data, performs data format decoding and manages the user interaction. It also acts as interface to external devices like smart card readers, 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 presentation agent is only present on a terminal.

The **access agent** provides MHEG object and multimedia content location, access and communication services. It makes the location of the various objects (multimedia contents, MHEG objects, scripts, application specific data) transparent to its clients, i.e. the processes trying to access these objects. When an object is requested, the access agent is able first to locate it (possibly by issuing requests to directory services), then to retrieve it from local or distant storage (possibly by issuing requests to repository services and remote equipments), finally to provide access to it (possibly by using MHEG object and content encoding/decoding services). In the reverse way, the access agent also handles the forwarding of objects to remote equipments, their registration in directories and their storage in repositories.

The **local application** manages the logic of the application on 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 make use of a script execution service provided by a script processor. Scripts are parts of the application which are interchanged during the course of an application. A script processor is a functional unit able to execute scripts, it may be the script itself (if interchanged in executable form), a script language interpreter or an MHEG Script Interchange Representation (SIR) interpreter.

The **MHEG engine** provides an MHEG interpretation service. It 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 **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 multimedia content presentation service provided by a presentation agent.

The **MHEG-API** allows MHEG applications to access the MHEG interpretation service provided by an MHEG engine. The MHEG-API client application may be either the local application, or it may run on a remote device. If the application is running on a remote device, it may access the MHEG-API using the end-to-end protocol "application – distant MHEG engine" (9). The MHEG-API is specified in this Recommendation.

The **access API** allows processes (e.g. local application, script interpreter, MHEG engine, presentation agent) to access the MHEG object and multimedia content location, access and communication services provided by an access agent.

The **service support unit** 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 Videoconferencing service. The functionality handled by the service support unit is service-specific.

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 terminal-to-host configurations.

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** "**access agent – access agent**" enables the handling, maintenance and exchange of objects and data in a distributed environment.

5.1.2 The MHEG-API

This subclause describes the terminology applicable to the MHEG-API.

The MHEG-API is the interface through which an MHEG application is allowed to control an MHEG engine.

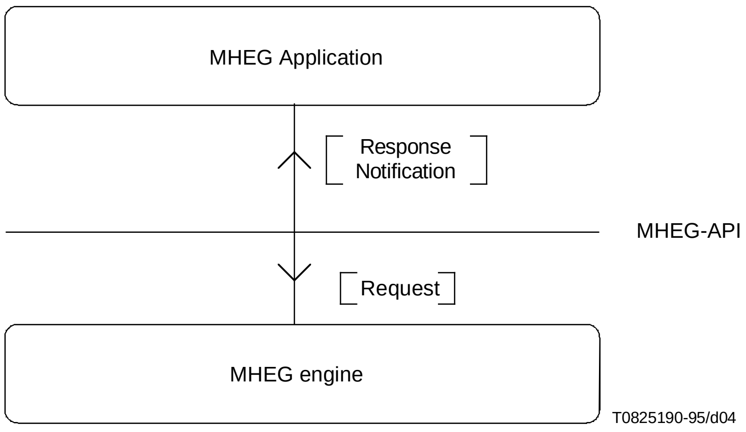


Figure 4/T.174 – The MHEG-API

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.

The **MHEG-API** gives access to **MHEG interpretation** and (optionally) **MHEG object access** 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 software services.

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:

– requests that require no response are called **asynchronous requests**;

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

– 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.

5.2 Functional specification of the MHEG-API

5.2.1 MHEG usage specifications

This subclause introduces MHEG usage rules. They consist of clarifications and interpretations of the MHEG standard regarding the definition of the main entities addressed by the MHEG standard that may be handled in MHEG using applications.

A clear understanding of the MHEG-related entities implies a formal definition of the identity of the entities that are handled and/or transformed by components of an MHEG using application, as well of how they can be identified or referenced within different components.

5.2.1.1 Definitions

These subclauses introduce the concepts of identity, identification and referencing. The following precisions bring additional semantics with regard to the MHEG standard (which makes no clear distinction between reference and identification), and are applicable as definitions throughout this Recommendation.

The identity of an object is itself. The identity function is defined by Identity(A) = A. Any object has an identity. Objects A and B have the same identity if, and only if, A and B are the same object.

The identification of an object is an unambiguous way to determine its identity. If objects A and B have the same identifier, then they have the same identity, i.e. they are the same object. The identifier is often, but not necessarily, contained in the object. Objects may have several identification modes (with different types of identifier), though this is neither a useful nor recommended policy. If an object has no identifier, then it cannot be identified, although it has an identity.

Referencing an object is a convenient way to associate a name with an object. To determine the object which is referenced, the object reference has to be resolved by some process. One reference may only reference one object at a given time, but one object may be referenced in many ways. Unlike its identifier, references to an object are therefore not guaranteed to be unique. Moreover, it is possible to reference objects which otherwise are not identifiable.

5.2.1.2 MHEG objects

According to the MHEG standard, an MHEG object is defined as a coded representation. Therefore, MHEG objects are bitstrings. The identity of an MHEG object is its bitstring. MHEG objects are 'form a' objects as described in the MHEG standard, 6.2.4. MHEG object A and MHEG object B are identical if and only if they **are** the same sequence of bits.

An MHEG object is not a physical object, but rather an abstraction (a specified sequence of bits) which may have many representations (i.e. different objects) of different types: interchanged MHEG objects, stored MHEG objects, mh-objects, etc. Such representations are handled by different software services.

An MHEG object may be identified by an MHEG identifier. MHEG identifiers are the **only** way to identify MHEG objects. The structure and coded representation of MHEG identifiers is defined by the MHEG standard. The MHEG identifier of an MHEG object must be encoded inside the MHEG object. Since the attribute is optional, some MHEG objects do not have an MHEG identifier. Such MHEG objects cannot be identified. The MHEG standard imposes a constraint on the design of MHEG-using applications which is that MHEG object A and MHEG object B shall not have the same MHEG identifier unless they are identical.

The MHEG generic reference describes all possible ways to reference an MHEG object.

5.2.1.3 Mh-objects

An mh-object is an internal representation of an MHEG object within a process or system. An mh-object is not an MHEG object. Within an MHEG engine, mh-objects represent 'available' MHEG objects. Mh-objects are 'form b' objects as described in the MHEG standard, 6.2.4. An mh-object represents one MHEG object, i.e. there is always a bitstring that corresponds to an mh-object. An MHEG engine shall not handle more than one mh-object to represent one MHEG object.

As a consequence, mh-objects handled by MHEG engines may be identified using MHEG identifiers. In addition, other mechanisms for identifying mh-objects (e.g. symbolic identification) may be defined by the application, provided their internal representation allows for it. This is especially useful when some of the MHEG objects represented by an MHEG engine's mh-objects are non-identifiable, i.e. have no MHEG identifier. This allows to guarantee that all mh-objects shall be identifiable.

Mh-objects are referenced the same way MHEG objects are. References to MHEG objects for which the MHEG engine handles an mh-object will usually be resolved by addressing this mh-object.

5.2.1.4 Rt-objects

An rt-object is a run-time 'instance' (or copy) of a 'model' mh-object, which is created and handled by an MHEG engine in the purpose of presentation. An rt-object is not an MHEG object. Within an MHEG engine, rt-objects represent 'rt-available' MHEG objects. Rt-objects are 'form c' objects as described in the MHEG standard, 6.2.4. There may be none or several rt-objects which are 'presentable' copies of one mh-object. An rt-object always has exactly one mh-object as its model.

Rt-objects may be identified using rt-object identifiers whose 'model object identification' part is an MHEG identifier. The structure and coded representation of rt-object identifiers are defined by the MHEG standard. In addition, other mechanisms for identifying mh-objects (e.g. symbolic identification) may be defined by the application, provided their internal representation allows for it. This is especially useful when some of the MHEG objects represented by an MHEG engine's mh-objects used as models for rt-objects are non-identifiable, i.e. have no MHEG identifier. This allows to guarantee that all rt-objects shall be identifiable.

Rt-objects may be referenced using MHEG generic references.

5.2.1.5 Channels

Channels are objects defined by the MHEG standard and handled by the MHEG engines. They may be identified using channel identifiers.

5.2.1.6 Interchanged MHEG objects

Interchanged MHEG objects are representations of MHEG objects which are being communicated at a given point in time using a network or storage medium. One given MHEG object (i.e. bitstring) may be interchanged many times between many places, i.e. represented by many interchanged MHEG objects. An MHEG external identifier may identify an interchanged MHEG object, and therefore reference an MHEG object through its location and time of interchange. However, it should be noted that an MHEG external identifier may not actually identify an MHEG object.

Stored MHEG objects are representations of MHEG objects which are usually located in files or database records. For instance, one given MHEG object (i.e. bitstring) may be stored in many places, i.e. represented by many stored MHEG objects. Such locations are usually identified using file names or database identifiers. An MHEG external identifier may identify a storage location for an MHEG object, and therefore reference an MHEG object through its storage location.

5.2.2 Description of MHEG-related services

This subclause introduces the concept of MHEG-related services, i.e. common use software services allowing to handle MHEG-related entities. These services are provided by the building blocks (components) of the MHEG using application architecture upon which the MHEG application is built. Clients and services do not necessarily know each other and may be related through a mediating entity. In this building blocks approach, implementations of the services make use of each other.

An MHEG application may make use of some or all of these following MHEG-related services:

– the **MHEG interpretation service** allows to control the behaviour of an MHEG engine, i.e. the interpretation and presentation of MHEG objects;

– the **MHEG object access service** allows to access and modify the attributes of 'logical' MHEG objects;

– the **MHEG object communication service** allows to transfer MHEG objects between locations;

- the **MHEG object location service** allows to manage and resolve references to MHEG objects;
- the **MHEG object handling service** allows to manage access to and interchange of physical MHEG objects;
- the **MHEG object storage service** allows to store MHEG object bitstrings and access them;
- the **MHEG object encoding/decoding service** allows to encode or decode MHEG objects;
- additional services for real-time distribution, presentation and production of MHEG objects and contents may be considered.

Different kinds of MHEG-related entities, bearing strong relationships with each other but being nevertheless of different types (therefore not comparable), are being handled by these software services:

- the MHEG interpretation service handles rt-objects, sockets, channels and mh-objects;
- the MHEG object access service handles mh-objects;
- the MHEG object handling service 'handles' MHEG objects. Since MHEG objects are virtual rather than physical objects, this service relies on services that handle physical 'representations' of MHEG objects, such as storage or transport services;
- the MHEG object location service handles MHEG generic references. Through the use of maps, it is able to resolve such references, i.e. translate them into identifiers understandable by the requesting application;
- the MHEG object communication service handles interchanged MHEG objects;
- the MHEG object storage service handles stored MHEG objects and manages MHEG object storage locations, whose type depend on the underlying storage mechanism, e.g. files, database records;
- the MHEG object encoding/decoding service provides functions for transforming mh-objects into MHEG objects and vice versa.

The MHEG-API consists of the interface provided by the following services, that shall be provided by Conforming MHEG engines:

- the MHEG interpretation service (mandatory);
- the MHEG object access service (optional).

6 API definition principles

6.1 Satisfaction of technical requirements on the MHEG-API

Following the recommendations of ISO/IEC JTC 1 N 2965 [11]: "Guidelines for JTC 1 API Standardisation", the MHEG-API is defined as an abstract API specification, i.e. a language-independent description of the semantics of a set of functionality in an abstract syntax using abstract data types.

Following the recommendations of ETR 225, the MHEG-API standard should meet the following requirements:

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

The **portability** requirement states that 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.

This Recommendation meets the portability requirement by the definition of an abstract API specification.

The **genericity** requirement states that the MHEG-API standard should provide appropriate support to cover all the common requirements of MHEG applications.

This Recommendation meets the genericity requirement through defining the MHEG-API at the most basic level, e.g. by defining primitives that match MHEG elementary actions and data types that match MHEG data types. This guarantees to maximise the range of MHEG object manipulations made available to applications.

The **conformance testability** requirement states that 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 as well as the conformance of MHEG applications to the MHEG-API standard, i.e. the correct use of this API by an MHEG application under test.

This Recommendation meets the conformance testability requirement by formal expression of the requirements on conforming implementations and conforming applications, as well as by the use of a formal description technique for the definition of the MHEG-API.

The **implementability** requirement states that 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.

This Recommendation meets the implementability requirement by the provision of informative guidelines to deduce language binding specifications and message encoding rules from the abstract API specification.

6.2 Use of Interface Definition Language (IDL) ISO/IEC 14750 [10]

The MHEG-API is defined using IDL.

6.2.1 Comprehensive introduction to IDL

IDL is a formal description technique for specifying the services provided by objects for use by applications or other objects. 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 Recommendation 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.

6.2.2 The Interface Definition Language

This subclause describes the main concepts that are necessary to the understanding of the MHEG-API definition.

The object model provides an organised representation of objects concepts and terminology. It defines a partial model for computation that embodies the key characteristics of objects as realised by the presented technologies.

The object model presented in this Recommendation is abstract in that it is not directly realised by any particular technology.

An objects system provides services to its clients. A client of a service is any entity capable of requesting the service.

6.2.2.1 Objects

An object system includes entities known as objects. An object is an identifiable, encapsulated entity that provides one or more services that can be requested by a client.

6.2.2.2 Requests

Clients request services by issuing requests. A request is an event, i.e. something that occurs at a particular time. The information associated with a request consists of an operation, a target object, zero or more (actual) parameters, and an optional request context.

A *request form* is a description or pattern that can be evaluated or performed multiple times to cause the issuing of requests.

A *value* is anything that may be a legitimate (actual) parameter in a request. A value may identify an object, for the purpose of performing the request. A value that identifies an object is called object name.

An *object reference* that reliably denotes a particular object. Specifically, an object reference will identify the same object each time the reference is used in a request (subject to certain pragmatic limit in space and time). An object may be denoted by multiple, distinct object references.

A request may have parameters that are used to pass data to the target object; it may also have a request context which provides additional information about the request.

A request causes a service to be performed on behalf of the client. One outcome of performing a service is returning to the client the results, if any, defined for the request.

If an abnormal condition occurs during the performance of a request, an *exception* is returned. The exception may carry additional return parameters particular to the exception.

The request parameters are identified by position. A parameter may be an input parameter, an output parameter, or an input-output parameter. A request may also return a single result value, as well as any output parameters.

The following semantics holds for all requests:

– any aliasing of parameter values is neither guaranteed removed nor guaranteed preserved;

– the order in which aliased output parameters are written is not guaranteed;

– any output parameters are undefined if an exception is returned;

– the values which may be returned in an input-output parameter may be constrained by the value which was input.

6.2.2.3 Types

A type is an identifiable with an associated predicate (a single-argument mathematical function with a boolean result) defined over values. A value satisfies a type if the predicate is true for that value. A value that satisfies a type is called *member of the type*.

Types are used in signatures to restrict a possible parameter or to characterise a possible result.

The *extension* of the type is the set of values that satisfy the type at any particular time.

An object type is a type whose members are objects (literally, values that identify objects). In other words, an object type is satisfied only by (values that identify) objects.

6.2.2.4 Interfaces

An *interface* is a description of a set of possible operations that a client may request of an object. An object satisfies an interface if it can be specified as target object in each potential request described by the interface.

An *interface type* is a type that is satisfied by any object (literally, any value that identifies an object) that satisfies a particular interface.

6.2.2.5 Operations

An *operation* is an identifiable entity that denotes a service that can be requested.

An operation is identified by an *operation identifier*. An operation is not a value.

An operation has a signature that describes the legitimate values of request parameters and returned results. In particular a signature consists of:

- a specification of parameters required in requests for that operation;
- a specification of the results of the operations;
- a specification of the exceptions that may be raised by a request for the operation and the types of parameters accompanying them;
- a specification of additional contextual information that may affect the request;
- an indication of the execution semantics the client should expect from a request for the operation.

A *parameter* is characterised by its mode and its type. The mode indicates whether the value should be passed from the client to the server (in), from the server to the client (out), or both (in-out). The parameter's type constrains the possible value which may be passed in the direction(s) dictated by the mode.

The *return result* is a distinguished out parameter.

An *exception* is an indication that an operation request was not performed successfully. An exception may be accompanied by additional exception-specific information.

A *request context* provides additional, operation-specific information that may effect the performance of a request.

Two styles of *execution semantics* are defined by the object model:

- At-most-once: If an operation request returns successfully, it was performed exactly once; if it returns an exception indication, it was performed at most once.
- Best-effort: A best-effort operation is a request-only operation, i.e. it cannot return any results and the requester never synchronises with the completion, if any, of the request.

The execution semantics to be expected are associated with an operation. This prevents a client and object implementation from assuming different execution semantics.

Note that the client is able to invoke an at-most-once operation in a synchronous or deferred-synchronous manner.

6.2.2.6 Attributes

An interface may have attributes. An attribute is logically equivalent to declaring a pair of accessor functions: one to retrieve the value of the attribute and one to set the value of the attribute.

An attribute may be read-only, in which case only the retrieval accessor function is defined.

6.2.2.7 Subtyping versus inheritance

Subtyping is a relationship between types based on their interfaces. It defines the rules by which objects of one type are determined to be acceptable in contexts expecting another type. Inheritance is a mechanism for reuse. Many object systems do not distinguish between subtyping and inheritance. The following subclause defines the two concepts separately, but then explicitly states how they are related.

6.2.2.8 Subtyping

The object model supports subtyping for object types. Intuitively, one type is a subtype of another if the first is a specialisation or refinement of the second. Operationally it means that any object of the first type can be used in any context that expects an object of the second type; that is, if S is a subtype of T, an object of type S may be used wherever an object of type T may be used. In other words, objects of type S are also of type T. Subtypes can have multiple parent types, with the implication that an object that is an instance of type S is also an instance of all supertypes of type S. The relationships between types define a type hierarchy, which can be drawn as a directed acyclic graph.

6.2.2.9 Inheritance

Inheritance is a notational mechanism for defining a type S in terms of another type T. The definition of S inherits all the operations of T and may provide other operations. Intuitively, inherits means that the operations defined for T are also defined for or can be used by S. Subtyping is a relationship between interfaces (types). Inheritance can apply to both interfaces and implementations; that is both interfaces and implementations can be inherited. The object model is concerned with inheritance of interfaces. It does not specify what can happen with implementations of inherited operations (for example, whether they may be changed or overridden by a subtype).

6.2.3 Principles for mapping IDL interfaces to API primitives

The MHEG engine interface consists of a set of API primitives that can be organised into clusters according to the target entity of a primitive. 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 definition follows an object-oriented methodology without requiring the implementations to use object-oriented design or programming techniques.

Services are defined not as the interface that a module should provide but as a set of operations conspiring to offer a service, to which clients should have access.

The MHEG-API therefore consists of IDL interface objects which provide operations that map API primitives. The object instance on which an operation is requested corresponds to the main (target) parameter of the API primitive.

6.2.4 Fulfilment of technical requirements

The use of IDL contributes to the fulfilment of the portability and implementability technical requirements:

– IDL is independent from a programming language. Moreover, publicly available specifications for IDL language binding to C and C++ exist, and others are under study;

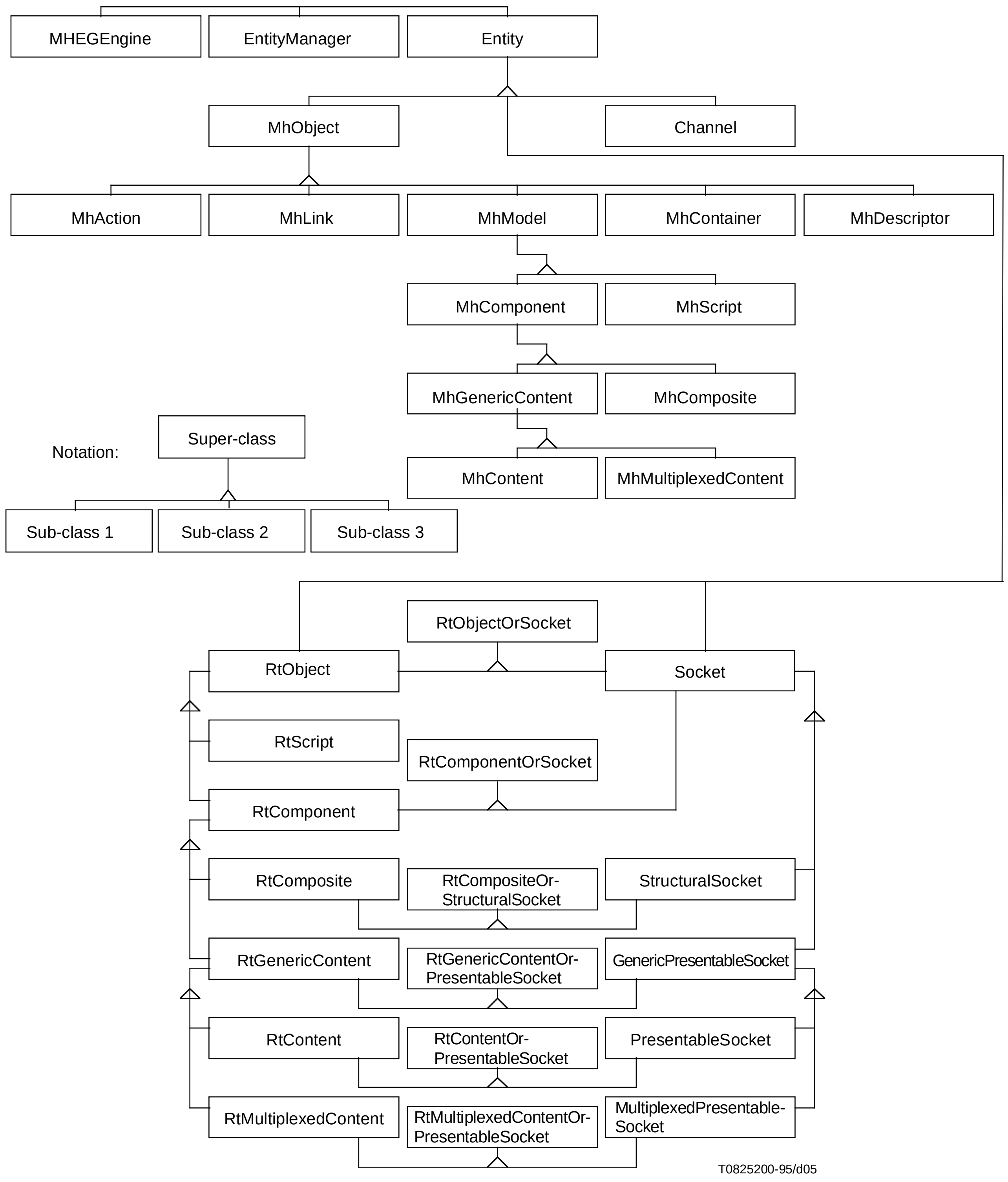
– IDL provides a complete formal description language which allows a very concise, readable and efficient specification of the MHEG-API. Moreover, this formal description language is also appropriate for automatic compilation, which means that MHEG-API implementations could be automatically generated 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.

6.3 Overview of the API definition and general principles

6.3.1 The MHEG-API Object model

This subclause presents the object model, i.e. the object types (interfaces) provided by the MHEG-API and their subtyping relationships.

It may be noted 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 – MhObjects (and their subtypes) match form b) objects as defined in ISO/IEC 13522-1: "Information technology – Coding of multimedia and hypermedia information" [1], 6.2.4, i.e. Objects available to the MHEG engine.

NOTE 2 – RtObjects (and their subtypes) match form c) objects as defined in ISO/IEC 13522-1: "Information technology – Coding of multimedia and hypermedia information" [1], 6.2.4, i.e. Instances of MhObjects available to the presentation process.

Figure 5/T.174 – Object model

7 Definition of the MHEG-API

7.1 Mandatory primitives

7.1.1 `MHEGEngine` object

The following subclause defines the operations of the `MHEGEngine` object.

7.1.1.1 `initialiseEngine` operation

Synopsis:

Interface:	`MHEGEngine`
Operation:	`initialiseEngine`
Result:	`void`

Description:

This operation performs any necessary initialisation of the interface. It shall be invoked before any other interface operations defined in this Recommendation are invoked. It can be invoked multiple times, in which case each invokation shall reinitialise the MHEG Engine.

7.1.1.2 `shutdownEngine` operation

Synopsis:

Interface:	`MHEGEngine`
Operation:	`shutdownEngine`
Result:	`void`

Description:

This operation deletes all service-generated interface objects associated with the current session. The next operation that shall be accepted by an MHEG engine is an `initialiseEngine` operation.

7.1.1.3 IDL description

```
interface MHEGEngine    {

     void    initialiseEngine();
     void    shutdownEngine();

};
```

7.1.2 `NotificationManager` object

The following subclause defines the operations of the `NotificationManager` object.

7.1.2.1 `getReturnability` operation

Synopsis:

Interface:	`NotificationManager`
Operation:	`getReturnability`
Result:	`sequence<unsigned short>`

Description:

This operation retrieves the returnability behaviour of the MHEG engine.

The operation returns a list of numbers identifying available notifications.

7.1.2.2 `getNotification` operation

Synopsis:

Interface:	`NotificationManager`	
Operation:	`getNotification`	
Result:	`void`	
In:	`unsigned short`	`notification_number`
Out:	`sequence<GenericValue>`	`values`
Out:	`sequence<MhObjectReference>`	`objects`
Exception:	`InvalidParameter`	

Description:

This operation retrieves a notification from the MHEG engine.

The `notification_number` parameter identifies the notification. This identification may be the result of a `getReturnability` operation.

The `values` parameter specifies the returned values.

The `objects` parameter specifies the returned object references.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.2.3 IDL description

```
interface    NotificationManager

     sequence<unsigned short>
         getReturnability();

     void
         getNotification(
            in unsigned short
                notification_number,
            out sequence<GenericValue>
                values,
            out sequence<MhObjectReference>
                objects)
     raises(InvalidParameter);

};
```

7.1.3 `EntityManager` object

The following subclause defines the operations of the `EntityManager` object.

7.1.3.1 `getAvailableMhObjects` operation

Synopsis:

Interface:	`EntityManager`
Operation:	`getAvailableMhObjects`
Result:	`sequence<MHEGIdentifier>`

Description:

This operation retrieves the mh-objects available to the MHEG engine.

An mh-object is either ’not ready’ (in period O1), ’processing’ (in period O2 or O4) or ’ready’ (in period O3). The operation retrieves those mh-objects which are in period O3.

The operation returns the identifiers of the available mh-objects.

7.1.3.2 `getAvailableRtObjects` operation

Synopsis:

Interface:	`EntityManager`
Operation:	`getAvailableRtObjects`
Result:	`sequence<RtObjectIdentifier>`

Description:

This operation retrieves the rt-objects available to the MHEG engine.

An rt-object is either ’not available’ (in period R1), ’processing’ (in period R2 or R4) or ’available’ (in period R3 and its subperiods). The operation retrieves those rt-objects which are in period R3.

The operation returns the identifiers of the available rt-objects.

7.1.3.3 `getAvailableChannels` operation

Synopsis:

Interface:	`EntityManager`	
Operation:	`getAvailableChannels`	
Result:	`sequence<ChannelIdentifier>`	

Description:

This operation retrieves the channels available to the MHEG engine.

A channel is either 'not available' (in period C1), 'processing' (in period C2 or C4) or 'available' (in period C3). The operation retrieves those channels which are in period C3.

The operation returns the identifiers of the available channels.

7.1.3.4 `releaseAlias` operation

Synopsis:

Interface:	`EntityManager`	
Operation:	`releaseAlias`	
Result:	`void`	
In:	`string`	`alias`
Exception:	`InvalidParameter`	

Description:

This operation enables to release an alias. It cancels the assignments of this alias to entities.

The `alias` parameter specifies the value of the released alias.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.3.5 IDL description

```
interface    EntityManager                          {

     sequence<MHEGIdentifier>
         getAvailableMhObjects();

     sequence<RtObjectIdentifier>
         getAvailableRtObjects();

     sequence<ChannelIdentifier>
         getAvailableChannels();

     void
         release Alias(
             in string
                 alias)
     raises(InvalidParameter);

};
```

7.1.4 `Entity` object

The following subclause defines the operations of the `Entity` object.

7.1.4.1 `setAlias` operation

Synopsis:

Interface:	`Entity`	
Operation:	`setAlias`	
Result:	`void`	
In:	`string`	`alias`
Exception:	`InvalidTarget`	

Description:

This operation enables the assignment of an alias to any entity.

The `setAlias` operation triggers the execution of the ’set alias’ elementary action with the bound entity as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 34.2.1.

The `alias` parameter specifies the value of the ’alias’ parameter of the ’set alias’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.4.2 `getAlias` operation

Synopsis:

Interface:	`Entity`
Operation:	`getAlias`
Result:	`string`
Exception:	`InvalidTarget`

Description:

This operation retrieves the alias assigned to an entity.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.4.3 IDL description

```
interface Entity  {

    void
        setAlias(
            in string
                alias)
    raises(InvalidTarget);

    string
        getAlias()
    raises(InvalidTarget);

};
```

7.1.5 `MhObject` object

The following subclause defines the operations of the `MhObject` object. The object inherits from the `Entity` object.

7.1.5.1 `bind` operation

Synopsis:

Interface:	`MhObject`	
Operation:	`bind`	
Result:	`MHEGIdentifier`	
In:	`MhObjectReference`	`mh_object_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation binds the MhObject instance (an interface object instance) with an MHEG object (an MHEG entity).

The `mh_object_reference` parameter specifies the reference of the MHEG object.

The operation returns the identifier of the bound MHEG object.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.5.2 `unbind` operation

Synopsis:

Interface:	`MhObject`	
Operation:	`unbind`	
Result:	`void`	
Exception:	`NotBound`	

Description:

This operation cancels the binding between the MhObject instance (an interface object instance) and an MHEG object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.5.3 `prepare` operation

Synopsis:

Interface:	`MhObject`	
Operation:	`prepare`	
Result:	`MHEGIdentifier`	
In:	`MhObjectReference`	`mh_object_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation enables the creation of an MHEG object from a model object by the MHEG engine.

The `prepare` operation triggers the execution of the 'prepare' elementary action targeted at a single MHEG object.

The effect of the action on its target and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 36.2.1.

The `mh_object_reference` parameter specifies a reference to an MHEG object.

This operation implicitly binds the MhObject instance (an interface object instance) with the new prepared MHEG object (an MHEG entity).

The operation returns the identifier of the new prepared MHEG object bound with the MhObject instance.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.5.4 `destroy` operation

Synopsis:

Interface:	`MhObject`	
Operation:	`destroy`	
Result:	`void`	
Exception:	`NotBound`	
Exception:	`InvalidTarget`	

Description:

This operation enables the removing of an MHEG object by the MHEG engine.

The `destroy` operation triggers the execution of the 'destroy' elementary action targeted at a single MHEG object.

The effect of the action on its target and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 36.2.2.

This operation implicitly cancels the binding between the MhObject instance (an interface object instance) and the new destroyed MHEG object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.5.5 `getPreparationStatus` operation

Synopsis:

Interface:	`MhObject`
Operation:	`getPreparationStatus`
Result:	`PreparationStatusValue`
Exception:	`NotBound`
Exception:	`InvalidTarget`

Description:

This operation retrieves the availability of an MHEG object to the MHEG engine.

The `getPreparationStatus` operation triggers the execution of the 'get preparation status' elementary action with the bound MHEG object as its single target.

The effect of the action on its target, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 36.3.1.

The operation returns the availability of the MHEG object bound with the MhObject instance. The returned value is either `NOT_READY`, `PROCESSING` or `READY`.

When the returned value is `NOT_READY`, the operation implicitly cancels the binding between the MhObject instance (an interface object instance) and the MHEG object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.5.6 `getIdentifier` operation

Synopsis:

Interface:	`MhObject`
Operation:	`getIdentifier`
Result:	`MHEGIdentifier`
Exception:	`NotBound`

Description:

This operation retrieves the identifier of the MHEG object (an MHEG entity) bound with the MhObject instance (an interface object instance).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.5.7 `kill` operation

Synopsis:

Interface:	`MhObject`
Operation:	`kill`
Result:	`void`

Description:

This operation deletes the MhObject instance (an interface object instance).

7.1.5.8 IDL description

```
interface    MhObject: Entity                      {

    MHEGIdentifier
        bind(
            in MhObjectReference
                mh_object_reference)
    raises(AlreadyBound, InvalidTarget);
    void
        unbind()

    raises(NotBound);

    MHEGIdentifier
        prepare(
            in MhObjectReference
                mh_object_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        destroy()
    raises(NotBound, InvalidTarget);

    PreparationStatusValue
        getPreparationStatus()
    raises(NotBound, InvalidTarget);

    MHEGIdentifier
        getIdentifier()
    raises(NotBound);

    void
        kill();

};
```

7.1.6 `MhAction` object

The following subclause defines the operations of the `MhAction` object. The object inherits from the `MhObject` object.

7.1.6.1 `delay` operation

Synopsis:

Interface:	`MhAction`	
Operation:	`delay`	
Result:	`void`	
In:	`unsigned short`	`nested_action_number`
In:	`unsigned long`	`delay`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation enables to delay the process of nested actions within the mh-action.

The `nested_action_number` parameter specifies the nested action after which the delay is to be processed.

The `delay` parameter specifies the duration of the delay expressed in Generic Temporal Unit (GTU).

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.6.1.2 IDL description

```
interface    MhAction: MhObject                          {

    void
        delay(
           in unsigned short
               nested_action_number,
           in unsigned long
               delay)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.7 `MhLink` object

The following subclause defines the operations of the `MhLink` object. The object inherits from the `MhObject` object.

7.1.7.1 `abort` operation

Synopsis:

Interface:	`MhLink`
Operation:	`abort`
Result:	`void`
Exception:	`InvalidTarget`

Description:

This operation aborts the processing of all the actions that have been activated by a link object. Each time the link condition is satisfied, the actions defining the link effect are activated and processed.

The `abort` operation triggers the execution of the 'abort' elementary action with the bound link object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 38.2.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.7.2 IDL description

```
interface MhLink: MhObject                                    {
    void
        abort()
    raises(InvalidTarget);
};
```

7.1.8 `MhModel` object

For the `MhModel` object no specific operations are defined. The object inherits from the `MhObject` object.

7.1.8.1 IDL description

```
interface MhModel: MhObject {};
```

7.1.9 `MhComponent` object

For the `MhComponent` object no specific operations are defined. The object inherits from the `MhModel` object.

7.1.9.1 IDL description

```
interface MhComponent: MhModel {};
```

7.1.10 `MhGenericContent` object

The following subclause defines the operations of the `MhGenericContent` object. The object inherits from the `MhComponent` object.

7.1.10.1 `copy` operation

Synopsis:

Interface:	`MhContent`	
Operation:	`copy`	
Result:	`void`	
In:	`sequence<MhObjectReference>`	`copies`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the copy of a content object "source" in a set of content objects "copies" or the copy of a multiplexed content object "source" in a set of multiplexed content objects "copies".

The `copy` operation triggers the execution of the ’copy’ elementary action with the bound content object or multiplexed content object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 40.2.1.

The `copies` parameter specifies the value of the ’copies’ parameter of the ’copy’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.10.2 IDL description

```
interface MhGenericContent: MhComponent                     {

    void
        copy(
            in sequence<MhObjectReference>
                  copies)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.11 `MhContent` object

The following subclause defines the operations of the `MhContent` object. The object inherits from the `MhGenericContent` object.

7.1.11.1 `setData` operation

Synopsis:

Interface:	`MhContent`	
Operation:	`setData`	
Result:	`void`	
In:	`boolean`	`substitution_indicator`
In:	`sequence<DataElement>`	`data_elements`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation allows to store or to modify the generic value in the data of a content object.

The `setData` operation triggers the execution of the ’set data’ elementary action with the bound content object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 39.2.1.

The `substitution_indicator` parameter specifies the value of the ’substitution indicator’ parameter of the ’set data’ action.

The `data_elements` parameter specifies the value of the ’data elements’ parameter of the ’set data’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.11.2 `getData` operation

Synopsis:

Interface:	`MhContent`	
Operation:	`getData`	
Result:	`GenericValue`	
In:	`sequence<long>`	`element_list_index`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation retrieves a generic value or an element of a generic list stored in the data of a content object.

The `getData` operation triggers the execution of the 'get data' elementary action with the bound content object as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 39.3.1.

The `element_list_index` parameter specifies the value of the 'element list index parameter' parameter of the 'get data' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.11.3 IDL description

```
interface MhContent: MhGenericContent                    {

    void
        setData(
            in boolean
               substitution_indicator,
            in sequence<DataElement>
                data_elements)
    raises(InvalidTarget, InvalidParameter);

    GenericValue
        getData(
            in sequence<long>
                element_list_index)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.12 `MhMultiplexedContent` object

The following subclause defines the operations of the `MhMultiplexedContent` object. The object inherits from the `MhGenericContent` object.

7.1.12.1 `setMultiplex` operation

Synopsis:

Interface:	`MhMultiplexedContent`	
Operation:	`setMultiplex`	
Result:	`void`	
In:	`sequence<StreamIdentifier>`	`stream_list`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the multiplexing of a list of content objects, the result is set in one multiplexed content object containing the multiplexed data.

The `setMultiplex` operation triggers the execution of the 'set multiplex' elementary action with the bound multiplexed content object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 41.2.1.

The `stream_list` parameter specifies the value of the ’stream list’ parameter of the ’set multiplex’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.12.2 `setDemultiplex` operation

Synopsis:

Interface:	`MhMultiplexedContent`	
Operation:	`setDemultiplex`	
Result:	`void`	
In:	`sequence<StreamIdentifier>`	`stream_list`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the demultiplexing of a multiple stream data of a multiplexed content object, e.g. a Moving Picture Experts Group (MPEG) stream; the result is set in a list of content objects which are generated if they do not exist yet. Each content object contains one demultiplexed stream.

The `setDemultiplex` operation triggers the execution of the ’set demultiplex’ elementary action with the bound multiplexed content object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 41.2.2.

The `stream_list` parameter specifies the value of the ’stream list’ parameter of the ’set demultiplex’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.12.3 IDL description

```
interface MhMultiplexedContent: MhGenericContent            {

    void
        setMultiplex(
           in sequence<StreamIdentifier>
               stream_list)
    raises(InvalidTarget, InvalidParameter);

    void
        setDemultiplex(
           in sequence<StreamIdentifier>
               stream_list)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.13 `MhComposite` object

For the `MhComposite` object no specific operations are defined. The object inherits from the `MhComponent` object.

7.1.13.1 IDL description

```
interface MhComposite: MhComponent {};
```

7.1.14 `MhScript` object

For the `MhScript` object no specific operations are defined. The object inherits from the `MhModel` object.

7.1.14.1 IDL description

```
interface MhScript: MhModel {};
```

7.1.15 `MhContainer` object

For the `MhContainer` object no specific operations are defined. The object inherits from the `MhObject` object.

7.1.15.1 IDL description

```
interface MhContainer: MhObject {};
```

7.1.16 `MhDescriptor` object

For the `MhDescriptor` object no specific operations are defined. The object inherits from the `MhObject` object.

7.1.16.1 IDL description

```
interface MhDescriptor: MhObject {};
```

7.1.17 `RtObjectOrSocket` object

The following subclause defines the operations of the `RtObjectOrSocket` object.

7.1.17.1 `setGlobalBehaviour` operation

Synopsis:

Interface:	`RtObject`	
Operation:	`setGlobalBehaviour`	
Result:	`void`	
In:	`GlobalBehaviour`	`global_behaviour`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation enables the modification of the global behaviour of an rt-object or a socket.

The `setGlobalBehaviour` operation triggers the execution of the ’set global behaviour’ elementary action with the bound rt-object or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 44.2.1.

The `global_behaviour` parameter specifies the value of the ’global behaviour’ parameter of the ’set global behaviour’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.17.2 `getGlobalBehaviour` operation

Synopsis:

Interface:	`RtObject`
Operation:	`getGlobalBehaviour`
Result:	`GenericValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves all the attributes value composing the global behaviour of each rt-object or socket to the MHEG engine.

The `getGlobalBehaviour` operation triggers the execution of the 'get global behaviour' elementary action with the bound rt-object or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 44.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.17.3 `run` operation

Synopsis:

Interface:	`RtObject`
Operation:	`run`
Result:	`void`
Exception:	`InvalidTarget`

Description:

This operation enables the activation of the rt-object or the socket by the running process.

The `run` operation triggers the execution of the 'run' elementary action with the bound rt-object or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 45.2.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.17.4 `stop` operation

Synopsis:

Interface:	`RtObject`
Operation:	`stop`
Result:	`void`
Exception:	`InvalidTarget`

Description:

This operation removes the rt-object from the running process.

The `stop` operation triggers the execution of the 'stop' elementary action with the bound rt-object as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 45.2.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.17.5 IDL description

```
interface RtObjectOrSocket                                {

    void
        setGlobalBehaviour(
            in GlobalBehaviour
                global_behaviour)
    raises(InvalidTarget, InvalidParameter);

    GenericValue
        getGlobalBehaviour()
    raises(InvalidTarget);

    void
        run()
    raises(InvalidTarget);

    void
        stop()
    raises(InvalidTarget);

};
```

7.1.18 `RtObject` object

The following subclause defines the operations of the `RtObject` object. The object inherits from the `RtObjectOrSocket` and from the `Entity` object.

7.1.18.1 `bind` operation

Synopsis:

Interface:	`RtObject`	
Operation:	`bind`	
Result:	`RtObjectIdentifier`	
In:	`RtObjectReference`	`rt_object_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation binds the RtObject instance (an interface object instance) with an rt-object (an MHEG entity).

The `rt_object_reference` parameter specifies the reference of the rt-object.

The operation returns the identifier of the bound rt-object.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.18.2 `unbind` operation

Synopsis:

Interface:	`RtObject`
Operation:	`unbind`
Result:	`void`
Exception:	`NotBound`

Description:

This operation cancels the binding between the RtObject instance (an interface object instance) and an rt-object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.18.3 `new` operation

Synopsis:

Interface:	`RtObject`	
Operation:	`new`	
Result:	`RtObjectIdentifier`	
In:	`RtObjectReference`	`rt_object_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation enables the creation of an rt-object from a model object by the MHEG engine.

The `new` operation triggers the execution of the ’new’ elementary action targeted at a single rt-object.

The effect of the action on its target and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 43.2.1.

The `rt_object_reference` parameter specifies a reference to an rt-object.

This operation implicitly binds the RtObject instance (an interface object instance) with the new created rt-object (an MHEG entity).

The operation returns the identifier of the new created rt-object bound with the RtObject instance.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.18.4 `delete` operation

Synopsis:

Interface:	`RtObject`
Operation:	`delete`
Result:	`void`
Exception:	`NotBound`
Exception:	`InvalidTarget`

Description:

This operation enables the removing of an rt-object by the MHEG engine.

The `delete` operation triggers the execution of the 'delete' elementary action targeted at a single rt-object.

The effect of the action on its target and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 43.2.2.

This operation implicitly cancels the binding between the RtObject instance (an interface object instance) and the new deleted rt-object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.18.5 `getAvailabilityStatus` operation

Synopsis:

Interface:	`RtObject`
Operation:	`getAvailabilityStatus`
Result:	`RtAvailabilityStatusValue`
Exception:	`NotBound`
Exception:	`InvalidTarget`

Description:

This operation retrieves the availability of an rt-object to the MHEG engine.

The `getAvailabilityStatus` operation triggers the execution of the 'get rt-availability status' elementary action with the bound rt-object as its single target.

The effect of the action on its target, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 43.3.1.

The operation returns the availability of the rt-object bound with the RtObject instance. The returned value is either `NOT_AVAILABLE`, `PROCESSING` or `AVAILABLE`.

When the returned value is `NOT_AVAILABLE,` the operation implicitly cancels the binding between the RtObject instance (an interface object instance) and the rt-object (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.18.6 `getIdentifier` operation

Synopsis:

Interface:	`RtObject`
Operation:	`getIdentifier`
Result:	`RtObjectIdentifier`
Exception:	`NotBound`

Description:

This operation retrieves the identifier of the rt-object (an MHEG entity) bound with the RtObject instance (an interface object instance).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.18.7 `kill` operation

Synopsis:

Interface:	`RtObject`
Operation:	`kill`
Result:	`void`

Description:

This operation deletes the RtObject instance (an interface object instance).

7.1.18.8 `getRunningStatus` operation

Synopsis:

Interface:	`RtObject`
Operation:	`getRunningStatus`
Result:	`RunningStatusValue`
Exception:	`InvalidTarget`

Description:

This operation gets the activation of each rt-object and each socket by the MHEG engine.

The `getRunningStatus` operation triggers the execution of the 'get running status' elementary action with the bound rt-object or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 45.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.18.9 IDL description

```
interface RtObject: Entity                                  {

   RtObjectIdentifier
       bind(
            in RtObjectReference
                rt_object_reference)
   raises(AlreadyBound, InvalidTarget);

   void
      unbind()
   raises(NotBound);

   RtObjectIdentifier
       new(
          in RtObjectReference
              rt_object_reference)
   raises(AlreadyBound, InvalidTarget);

   void
      delete()
   raises(NotBound, InvalidTarget);

   RtAvailabilityStatusValue
       getAvailabilityStatus()
   raises(NotBound, InvalidTarget);

   RtObjectIdentifier
       getIdentifier()
   raises(NotBound);

   void
      kill();
```

```
    RunningStatusValue
        getRunningStatus()
    raises(InvalidTarget);
};
```

7.1.19 `Socket` object

The following subclause defines the operations of the `Socket` object. The object inherits from the `RtObjectOrSocket` and from the `Entity` object.

7.1.19.1 `bind` operation

Synopsis:

Interface:	`Socket`	
Operation:	`bind`	
Result:	`SocketIdentification`	
In:	`SocketReference`	`socket_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation binds the Socket instance (an interface object instance) with a socket (an MHEG entity).

The `socket_reference` parameter specifies the reference of the socket.

The operation returns the identifier of the bound socket.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.19.2 `unbind` operation

Synopsis:

Interface:	`Socket`
Operation:	`unbind`
Result:	`void`
Exception:	`NotBound`

Description:

This operation cancels the binding between the Socket instance (an interface object instance) and a socket (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.19.3 `getIdentifier` operation

Synopsis:

Interface:	`Socket`
Operation:	`getIdentifier`
Result:	`SocketIdentification`

Description:

This operation retrieves the identifier of the socket (an MHEG entity) bound with the Socket instance (an interface object instance).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.19.4 `kill` operation

Synopsis:

Interface:	`Socket`
Operation:	`kill`
Result:	`void`

Description:

This operation deletes the Socket instance (an interface object instance).

7.1.19.5 `plug` operation

Synopsis:

Interface:	`Socket`	
Operation:	`plug`	
Result:	`void`	
In:	`PlugIn`	`plug_in`
Exception:	`InvalidTarget`	

Description:

This operation enables a dynamism in the presentation and structure. This operation specifies the information to be plugged into a socket. This is used to obtain a different presentation or structure from the same composite object model.

The `plug` operation triggers the execution of the ’plug’ elementary action with the bound socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 48.2.1.

The `plug_in` parameter specifies the value of the ’plug in’ parameter of the ’plug’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.19.6 `setVisibleDurationPosition` operation

Synopsis:

Interface:	`Socket`	
Operation:	`setVisibleDurationPosition`	
Result:	`void`	
In:	`VisibleDurationPosition`	`visible_duration_position`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies within the perceptible duration of the parent the position where to attach the visible duration of a socket.

The `setVisibleDurationPosition` operation triggers the execution of the ’set visible duration position’ elementary action with the bound socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.2.4.

The `visible_duration_position` parameter specifies the value of the ’parent relative generic space temporal position’ parameter of the ’set visible duration position’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.19.7 `getVisibleDurationPosition` operation

Synopsis:

Interface:	`Socket`
Operation:	`getVisibleDurationPosition`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the visible duration position value of the socket within its parent relative generic space. This value is retrieved in relative generic temporal unit.

The `getVisibleDurationPosition` operation triggers the execution of the ’get VD position’ elementary action with the bound socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.7.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.19.8 IDL description

```
interface Socket: Entity, RtObjectOrSocket                  {

    SocketIdentification
        bind(
            in SocketReference
               socket_reference)
    raises(AlreadyBound, InvalidTarget);

    void   unbind()
        raises(NotBound);

    SocketIdentification
        getIdentifier();

    void
        kill();

    void
        plug(
            in PlugIn
               plug_in)
    raises(InvalidTarget);

    void
        setVisibleDurationPosition(
            in VisibleDurationPosition
               visible_duration_position)
    raises(InvalidTarget, InvalidParameter);

    unsigned long
        getVisibleDurationPosition()
    raises(InvalidTarget);

};
```

7.1.20 `RtScript` object

The following subclause defines the operations of the `RtScript` object. The object inherits from the `RtObject` object.

7.1.20.1 `setParameters` operation

Synopsis:

Interface:	`RtScript`	
Operation:	`setParameters`	
Result:	`void`	
In:	`sequence<Parameter>`	`parameters`
Exception:	`InvalidTarget`	

Description:

This operation enables to pass parameters between rt-scripts and other MHEG entities.

The `setParameters` operation triggers the execution of the ’set parameters’ elementary action with the bound rt-script as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 46.2.1.

The `parameters` parameter specifies the value of the ’parameters’ parameter of the ’set parameters’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.20.2 `getTerminationStatus` operation

Synopsis:

Interface:	`RtScript`
Operation:	`getTerminationStatus`
Result:	`TerminationStatusValue`
Exception:	`InvalidTarget`

Description:

This operation gets the process termination of each rt-script and by the script process.

The `getTerminationStatus` operation triggers the execution of the 'get termination status' elementary action with the bound rt-script as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 47.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.20.3 IDL description

```
interface RtScript: RtObject                                  {

    void
        setParameters(
              in sequence<Parameter>
                  parameters)
    raises(InvalidTarget);

    TerminationStatusValue
        getTerminationStatus()
    raises(InvalidTarget);

};
```

7.1.21 `RtComponentOrSocket` object

The following subclause defines the operations of the `RtComponentOrSocket` object.

7.1.21.1 `setRGS` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setRGS`	
Result:	`void`	
In:	`ChannelIdentifier`	`channel_identifier`
Exception:	`InvalidTarget`	

Description:

This operation assigns an rt-component or a socket to a Relative Generic Space (RGS).

The `setRGS` operation triggers the execution of the 'set RGS' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 50.2.1.

The `channel_identifier` parameter specifies the value of the 'channel identifier' parameter of the 'set RGS' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.2 `getRGS` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getRGS`	
Result:	`RGSValue`	
Exception:	`InvalidTarget`	

Description:

This operation retrieves the RGS assigned to an rt-component or to a socket.

The `getRGS` operation triggers the execution of the 'get RGS' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 50.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.3 `setOpacity` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setOpacity`	
Result:	`void`	
In:	`unsigned short`	`opacity_rate`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	

Description:

This operation assigns an opacity rate value to an rt-component or a socket.

The `setOpacity` operation triggers the execution of the 'set opacity' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.2.1.

The `opacity_rate` parameter specifies the value of the 'opacity rate' parameter of the 'set opacity' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set opacity' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.4 `setPresentationPriority` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setPresentationPriority`	
Result:	`void`	
In:	`PresentationPriority`	`presentation_priority`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the presentation priority between the rt-components or sockets assigned to the same RGS. The operation defines the priority of the rt-component or socket with respect to the other rt-components or sockets assigned to the same RGS.

The `setPresentationPriority` operation triggers the execution of the 'set Presentation priority' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.2.2.

The `presentation_priority` parameter specifies the value of the 'presentation priority' parameter of the 'set Presentation priority' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set Presentation priority' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.5 `getOpacity` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getOpacity`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the opacity value of an rt-component or a socket.

The `getOpacity` operation triggers the execution of the 'get opacity' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.6 `getEffectiveOpacity` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getEffectiveOpacity`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the effective opacity value of an rt-component or a socket.

The `getEffectiveOpacity` operation triggers the execution of the 'get effective opacity' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.3.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.7 `getPresentationPriority` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getPresentationPriority`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the presentation priority of an rt-component or a socket.

The `getPresentationPriority` operation triggers the execution of the 'get presentation priority' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.3.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.8 `setVisibleDuration` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setVisibleDuration`	
Result:	`void`	
In:	`TemporalPosition`	`initial_temporal_position`
In:	`TemporalPosition`	`terminal_temporal_position`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation retrieves the presentation priority of an rt-component or a socket.

The `setVisibleDuration` operation triggers the execution of the 'set visible duration' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 51.3.3.

The `initial_temporal_position` parameter specifies the value of the 'initial temporal position' parameter of the 'set visible duration' action.

The `terminal_temporal_position` parameter specifies the value of the 'terminal temporal position' parameter of the 'set visible duration' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.9 `setTemporalTermination` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setTemporalTermination`	
Result:	`void`	
In:	`TemporalTermination`	`temporal_termination`
Exception:	`InvalidTarget`	

Description:

This operation specifies the type of temporal termination when the current temporal position passes the terminal temporal position.

The `setTemporalTermination` operation triggers the execution of the 'set temporal termination' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.2.2.

The `temporal_termination` parameter specifies the value of the 'temporal termination' parameter of the 'set temporal termination' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.10 `setCurrentTemporalPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setCurrentTemporalPosition`	
Result:	`void`	
In:	`TemporalPosition`	`temporal_position`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies a current temporal position within the VD.

The `setCurrentTemporalPosition` operation triggers the execution of the ’set current temporal position’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.2.3.

The `temporal_position` parameter specifies the value of the ’temporal position’ parameter of the ’set current temporal position’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.11 `setSpeed` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setSpeed`	
Result:	`void`	
In:	`Speed`	`the_speed`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation defines the speed of the presentation of an rt-component or socket. The effective presentation speed is calculated from this value by the MHEG engine.

The `setSpeed` operation triggers the execution of the ’set speed’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.2.5.

The `the_speed` parameter specifies the value of the ’speed’ parameter of the ’set speed’ action.

The `transition_duration` parameter specifies the value of the ’transition duration’ parameter of the ’set speed’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.12 `setTimestones` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setTimestones`	
Result:	`void`	
In:	`sequence<Timestone>`	`timestones`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies a complete set of temporal markers within the perceptible duration of the presentable.

The `setTimestones` operation triggers the execution of the 'set timestones' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.2.6.

The `timestones` parameter specifies the value of the 'timestones' parameter of the 'set timestones' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.13 `getInitialTemporalPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getInitialTemporalPosition`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the initial temporal position value of the rt-component or socket. This value is retrieved in Original generic space Generic Temporal Unit (OGTU).

The `getInitialTemporalPosition` operation triggers the execution of the 'get initial temporal position' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.14 `getTerminalTemporalPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getTerminalTemporalPosition`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the terminal temporal position value of the rt-component or socket. This value is retrieved in OGTU.

The `getTerminalTemporalPosition` operation triggers the execution of the ’get terminal temporal position’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.15 `getVDLength` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getVDLength`	
Result:	`unsigned long`	
In:	`GTIndicator`	`gt_indicator`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the VD length value of the rt-component or socket either in OGTU or in Relative generic space Generic Temporal Unit (RGTU).

The `getVDLength` operation triggers the execution of the ’get VD length’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.4.

The `gt_indicator` parameter specifies the value of the ’GT indicator’ parameter of the ’get VD length’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.16 `getTemporalTermination` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getTemporalTermination`
Result:	`TemporalTermination`
Exception:	`InvalidTarget`

Description:

This operation retrieves the ’temporal termination’ value of the rt-component or socket.

The `getTemporalTermination` operation triggers the execution of the ’get temporal termination’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.5.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.17 `getCurrentTemporalPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getCurrentTemporalPosition`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the current temporal position value of the rt-component or socket. This value is retrieved in OGTU.

The `getCurrentTemporalPosition` operation triggers the execution of the 'get current temporal position' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.6.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.18 `getSpeedRate` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getSpeedRate`
Result:	`short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the speed rate value of the rt-component or socket. This value is a percentage negative or positive. This speed rate is used to indicate the required change of speed since the Initial Original generic space Generic Temporal Ratio (IOGTR) and also the required direction of the presentation.

The `getSpeedRate` operation triggers the execution of the 'get speed rate' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.8.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.19 `getOGTR` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getOGTR`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the Original generic space Generic Temporal Ratio (OGTR) value of the rt-component or socket. This value is a positive or null numeric which corresponds to the number of OGTU to be mapped in one second.

The `getOGTR` operation triggers the execution of the 'get OGTR' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.9.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.20 `getEffectiveSpeedRate` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getEffectiveSpeedRate`
Result:	`short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the effective speed rate value of the rt-component or socket. This value is a percentage negative or positive. This effective speed rate is used to calculate the effective change of speed since the IOGTR and also the effective direction of the presentation.

The `getEffectiveSpeedRate` operation triggers the execution of the ’get effective speed rate’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.10.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.21 `getEffectiveOGTR` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getEffectiveOGTR`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the effective OGTR value of the rt-component or socket. This value is a positive or null numeric which corresponds to the effective number of OGTU to be mapped in one second.

The `getEffectiveOGTR` operation triggers the execution of the ’get effective OGTR elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.11.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.22 `getTimestoneStatus` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getTimestoneStatus`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the timestone status value of the rt-component or socket.

The `getTimestoneStatus` operation triggers the execution of the ’get timestone status’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 52.3.12.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.23 `setPerceptibleSizeProjection` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setPerceptibleSizeProjection`	
Result:	`void`	
In:	`PerceptibleSizeProjection`	`perceptible_size_projection`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation defines the projection of the perceptible size in its RGS.

The `setPerceptibleSizeProjection` operation triggers the execution of the 'set perceptible size projection' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.1.

The `perceptible_size_projection` parameter specifies the value of the 'perceptible size projection' parameter of the 'set perceptible size projection' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set perceptible size projection' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.24 `setAspectRatio` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setAspectRatio`	
Result:	`void`	
In:	`AspectRatio`	`preserved`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether in performing the projection of an rt-component or socket in the Channel Generic Space (CGS) [through the chain of mappings Original Generic Space (OGS)-RGS], the ratio between the Perceptible Size (PS) in Original generic space Generic Spatial Unit (OGSU), i.e. the OGS lengths, and the projection in Channel Generic Space Unit (CGSU) of the 'size of the content information' is to be the same for each axis. In such case the aspect ratio is preserved.

The `setAspectRatio` operation triggers the execution of the 'set aspect ratio preserved' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.3.

The `preserved` parameter specifies the value of the 'preserved' parameter of the 'set aspect ratio preserved' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.25 `setVisibleSize` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setVisibleSize`	
Result:	`void`	
In:	`VSGS`	`the_vsgs`
In:	`VS`	`the_vs`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the Visible Size (VS), which defines which portion of the PS is perceived by the user.

The `setVisibleSize` operation triggers the execution of the ’set visible size’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.4.

The `the_vsgs` parameter specifies the value of the ’vsgs’ parameter of the ’set visible size’ action.

The `the_vs` parameter specifies the value of the ’vs’ parameter of the ’set visible size’ action.

The `transition_duration` parameter specifies the value of the ’transition duration’ parameter of the ’set visible size’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.26 `setVisibleSizesAdjustment` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setVisibleSizesAdjustment`	
Result:	`void`	
In:	`sequence<AdjustmentAxis>`	`set_of_axes`
In:	`AdjustmentPolicy`	`adjustment_policy`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	

Description:

This operation specifies the adjustment of a set of VSs on a same axis or on a set of axes. All the VSs to be adjusted needs to be assigned to the same CGS.

The `setVisibleSizesAdjustment` operation triggers the execution of the ’set visible sizes adjustment’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.5.

The `set_of_axes` parameter specifies the value of the ’set of axes’ parameter of the ’set visible sizes adjustment’ action.

The `adjustment_policy` parameter specifies the value of the ’adjustment policy’ parameter of the ’set visible sizes adjustment’ action.

The `transition_duration` parameter specifies the value of the ’transition duration’ parameter of the ’set visible sizes adjustment’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.27 `setBox` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setBox`	
Result:	`void`	
In:	`BoxConstants`	`box`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether the rt-component or the socket is presented with a box to show the perimeter of the VS.

The `setBox` operation triggers the execution of the 'set box' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.6.

The `box` parameter specifies the value of the 'box' parameter of the 'set box' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.28 `setDefaultBackground` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setDefaultBackground`	
Result:	`void`	
In:	`unsigned short`	`background`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies whether the areas within the VS of an rt-component or a socket which are not filled by the presentation process have to be considered as opaque or transparent.

The `setDefaultBackground` operation triggers the execution of the 'set default background' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.7.

The `background` parameter specifies the value of the 'background' parameter of the 'set default background' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set default background' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.29 `setAttachmentPoint` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setAttachmentPoint`	
Result:	`void`	
In:	`AttachmentPointType`	`type`
In:	`AttachmentPoint`	`positions`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies one of the following AP: Perceptible Size Attachment Point (PSAP), Visible Size Internal Attachment Point (VSIAP) or Visible Size External Attachment Point (VSEAP).

The `setAttachmentPoint` operation triggers the execution of the 'set attachment point' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.8.

The `type` parameter specifies the value of the 'type' parameter of the 'set attachment point' action.

The `positions` parameter specifies the value of the 'positions' parameter of the 'set attachment point' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.30 `setAttachmentPointPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setAttachmentPointPosition`	
Result:	`void`	
In:	`AttachmentPointType`	`type`
In:	`ReferenceType`	`vseap_reference_point`
In:	`Lengths`	`the_lengths`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies the position of the VSIAP relatively to the PSAP, or the position of the VSEAP in its RGS relatively to the origin or to another VSEAP.

The `setAttachmentPointPosition` operation triggers the execution of the 'set attachment point position' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.9.

The `type` parameter specifies the value of the 'type' parameter of the 'set attachment point position' action.

The `vseap_reference_point` parameter specifies the value of the 'vseap reference point' parameter of the 'set attachment point position' action.

The `the_lengths` parameter specifies the value of the 'lengths' parameter of the 'set attachment point position' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set attachment point position' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.31 `setVisibleSizesAlignment` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setVisibleSizesAlignment`	
Result:	`void`	
In:	`SizeBorder`	`size_border`
In:	`long`	`interval`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	

Description:

This operation specifies the alignment of a set of VSs. The VSs are aligned on a border and an interval between two VSs may be provided. All the VSs to be aligned needs to be assigned to the same CGS.

The `setVisibleSizesAlignment` operation triggers the execution of the 'set visible sizes alignment' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.10.

The `size_border` parameter specifies the value of the 'size border' parameter of the 'set visible sizes alignment' action.

The `interval` parameter specifies the value of the 'interval' parameter of the 'set visible sizes alignment' action.

The `transition_duration` parameter specifies the value of the 'transition duration' parameter of the 'set visible sizes alignment' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.32 `setMovingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setMovingAbility`	
Result:	`void`	
In:	`UserControls`	`moving_ability`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether the user is able to move or not to move the VS of the targets in their CGS.

The `setMovingAbility` operation triggers the execution of the 'set moving ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.11.

The `moving_ability` parameter specifies the value of the 'moving ability' parameter of the 'set moving ability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.33 `setResizingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setResizingAbility`	
Result:	`void`	
In:	`UserControls`	`resizing_ability`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether the user is able to resize or not the VS of the targets in their CGS.

The `setResizingAbility` operation triggers the execution of the 'set resizing ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.12.

The `resizing_ability` parameter specifies the value of the ’resizing ability’ parameter of the ’set resizing ability’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.34 `setScalingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setScalingAbility`	
Result:	`void`	
In:	`UserControls`	`scaling_ability`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether the user is able to scale or not the PS of the targets in their CGS.

The `setScalingAbility` operation triggers the execution of the ’set scaling ability’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.13.

The `scaling_ability` parameter specifies the value of the ’scaling ability’ parameter of the ’set scaling ability’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.35 `setScrollingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setScrollingAbility`	
Result:	`void`	
In:	`UserControls`	`scrolling_ability`
Exception:	`InvalidTarget`	

Description:

This operation specifies whether the user is able to scroll or not the PS through the VS of the targets in their CGS.

The `setScrollingAbility` operation triggers the execution of the ’set scrolling ability’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.14.

The `scrolling_ability` parameter specifies the value of the ’scrolling ability’ parameter of the ’set scrolling ability’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.36 `getGSR` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getGSR`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the Generic Spatial Ratio (GSR) value of the OGS of the rt-component or socket. This ratio defines the number of OGSU which are to be mapped in one Relative generic space Generic Spatial Unit (RGSU).

The `getGSR` operation triggers the execution of the 'get GSR' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.37 `getPS` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getPS`	
Result:	`SpecifiedPosition`	
In:	`GSIndicator`	`gs`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the PS value of the rt-component or socket either in OGSU or in RGSU.

The `getPS` operation triggers the execution of the 'get PS' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.2.

The `gs` parameter specifies the value of the 'gs' parameter of the 'get PS' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.38 `getAspectRatio` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getAspectRatio`
Result:	`AspectRatio`
Exception:	`InvalidTarget`

Description:

This operation retrieves the aspect ratio value of the PS of the rt-component or socket.

The `getAspectRatio` operation triggers the execution of the 'get aspect ratio' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.4.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.39 `getPSAP` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getPSAP`	
Result:	`SpecifiedPosition`	
In:	`PointType`	`point_type`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the PSAP value of the rt-component or socket.

The `getPSAP` operation triggers the execution of the ’get PSAP’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.5.

The `point_type` parameter specifies the value of the ’point type’ parameter of the ’get PSAP’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.40 `getVSGS` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getVSGS`
Result:	`VSGS`
Exception:	`InvalidTarget`

Description:

This operation retrieves the Visible Size Generic Space (VSGS) value of the rt-component or socket.

The `getVSGS` operation triggers the execution of the ’get VSGS’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.6.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.41 `getVS` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getVS`
Result:	`SpecifiedPosition`
Exception:	`InvalidTarget`

Description:

This operation retrieves the VS value of the rt-component or socket in Visible Size Generic Spatial Unit (VSGSU).

The `getVS` operation triggers the execution of the ’get vs’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.7.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.42 `getBox` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getBox`
Result:	`BoxConstants`
Exception:	`InvalidTarget`

Description:

This operation retrieves the visible size box value of the rt-component or socket.

The `getBox` operation triggers the execution of the ’get box’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.8.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.43 `getDefaultBackground` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getDefaultBackground`	
Result:	`unsigned short`	
Exception:	`InvalidTarget`	

Description:

This operation retrieves the default background value of the VS of the rt-component or socket.

The `getDefaultBackground` operation triggers the execution of the ’get default background’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.9.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.44 `getVSIAP` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getVSIAP`	
Result:	`SpecifiedPosition`	
In:	`PointType`	`point_type`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the VSIAP value of the rt-component or socket.

The `getVSIAP` operation triggers the execution of the ’get VSIAP’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.10.

The `point_type` parameter specifies the value of the ’point type’ parameter of the ’get VSIAP’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.45 `getVSIAPPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getVSIAPPosition`
Result:	`SpecifiedPosition`
Exception:	`InvalidTarget`

Description:

This operation retrieves the VSIAP position value of the rt-component or socket. This is used to position the VSIAP relatively to the PSAP.

The `getVSIAPPosition` operation triggers the execution of the ’get VSIAP position’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.11.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.46 `getVSEAP` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getVSEAP`	
Result:	`SpecifiedPosition`	
In:	`PointType`	`point_type`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the VSEAP value of the rt-component or socket.

The `getVSEAP` operation triggers the execution of the ’get VSEAP’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.12.

The `point_type` parameter specifies the value of the ’point type’ parameter of the ’get VSEAP’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.47 `getVSEAPPosition` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getVSEAPPosition`	
Result:	`SpecifiedPosition`	
In:	`ReferencePoint`	`reference_point`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the VSEAP position value of the rt-component or socket relatively to a reference point. This is used to position the VSEAP relatively to the PSAP.

The `getVSEAPPosition` operation triggers the execution of the ’get VSEAP position’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.13.

The `reference_point` parameter specifies the value of the ’reference point’ parameter of the ’get VSEAP position’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.48 `getMovingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getMovingAbility`
Result:	`UserControls`
Exception:	`InvalidTarget`

Description:

This operation retrieves the moving ability value of the rt-component or socket.

The `getMovingAbility` operation triggers the execution of the 'get moving ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.14.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.49 `getResizingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getResizingAbility`
Result:	`UserControls`
Exception:	`InvalidTarget`

Description:

This operation retrieves the resizing ability value of the rt-component or socket.

The `getResizingAbility` operation triggers the execution of the 'get resizing ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.15.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.50 `getScalingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getScalingAbility`
Result:	`UserControls`
Exception:	`InvalidTarget`

Description:

This operation retrieves the scaling ability value of the rt-component or socket.

The `getScalingAbility` operation triggers the execution of the 'get scaling ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.16.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.51 `getScrollingAbility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getScrollingAbility`	
Result:	`UserControls`	
Exception:	`InvalidTarget`	

Description:

This operation retrieves the scrolling ability value of the rt-component or socket.

The `getScrollingAbility` operation triggers the execution of the 'get scrolling ability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.17.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.52 `setSelectability` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setSelectability`	
Result:	`void`	
In:	`unsigned short`	`min_number_of_selections`
In:	`unsigned short`	`max_number_of_selections`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns a 'minimum number of selections required' value and a 'maximum number of selections required' value to an rt-component or a socket. The MHEG engine calculates the 'selectability' value of the rt-component or socket from these two values. The 'effective selectability' value of the rt-component or socket is also calculated by the MHEG engine from this 'selectability' value and the 'effective selectability' value of the parent of the rt-component or socket.

The `setSelectability` operation triggers the execution of the 'set selectability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.2.1.

The `min_number_of_selections` parameter specifies the value of the 'min number of selections' parameter of the 'set selectability' action.

The `max_number_of_selections` parameter specifies the value of the 'max number of selections' parameter of the 'set selectability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.53 `setSelectionStatus` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setSelectionStatus`	
Result:	`void`	
In:	`SelectionStatusValue`	`selection_state`
Exception:	`InvalidTarget`	

Description:

This operation assigns a value to the ’selection status’ of an rt-component or a socket.

The `setSelectionStatus` operation triggers the execution of the ’set selection status’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.2.2.

The `selection_state` parameter specifies the value of the ’selection state’ parameter of the ’set selection status’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.54 `setSelectionPresentationEffectResponsibility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setSelectionPresentationEffectResponsibility`	
Result:	`void`	
In:	`Responsibility`	`the_responsibility`
Exception:	`InvalidTarget`	

Description:

This operation assigns a value to the ’selection presentation effect responsibility’ of an rt-component or a socket. This attribute indicates if it is the MHEG engine or the author who is responsible for reflecting a new state of the rt-component or socket as its single target

The `setSelectionPresentationEffectResponsibility` operation triggers the execution of the ’set selection presentation effect responsibility’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.2.3.

The `the_responsibility` parameter specifies the value of the ’responsibility’ parameter of the ’set selection presentation effect responsibility’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.55 `getSelectability` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getSelectability`	
Result:	`void`	
Out:	`unsigned short`	`min_number_of_selections`
Out:	`unsigned short`	`max_number_of_selections`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the ’minimum number of selections required’ and the ’maximum number of selections required’. If the ’maximum number of selections required’ is equal to 0, the ’selectability’ of the rt-component or socket is ’not selectable’.

The `getSelectability` operation triggers the execution of the ’get selectability’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.1.

The `min_number_of_selections` parameter specifies the value of the ’min number of selections’ parameter of the ’get selectability’ action.

The `max_number_of_selections` parameter specifies the value of the 'max number of selections' parameter of the 'get selectability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.56 `getEffectiveSelectability` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getEffectiveSelectability`
Result:	`EffectiveSelectability`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'effective selectability' attribute value of the rt-component or socket.

The `getEffectiveSelectability` operation triggers the execution of the 'get effective selectability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.57 `getSelectionStatus` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getSelectionStatus`
Result:	`SelectionStatusValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'selection status' value of the rt-component or socket.

The `getSelectionStatus` operation triggers the execution of the 'get selection status' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.58 `getSelectionMode` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getSelectionMode`
Result:	`SelectionModeValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'selection mode' attribute value of the rt-component or socket.

The `getSelectionMode` operation triggers the execution of the 'get selection mode' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.4.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.59 `getSelectionPresentationEffectResponsibility` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getSelectionPresentationEffectResponsibility`
Result:	`Responsibility`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'selection presentation effect responsibility' attribute value of the rt-component or socket.

The `getSelectionPresentationEffectResponsibility` operation triggers the execution of the 'get selection presentation effect responsibility' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.6.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.60 `setModifiability` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setModifiability`	
Result:	`void`	
In:	`unsigned short`	`min_number_of_modifications`
In:	`unsigned short`	`max_number_of_modifications`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns a 'minimum number of modifications required' value and a 'maximum number of modifications required' value to an rt-component or a socket. The MHEG engine calculates the 'modifiability' value of the rt-component or socket from these two values. The 'effective modifiability' value of the rt-component or socket is also calculated by the MHEG engine from this 'modifiability' value and the 'effective modifiability' value of the parent of the rt-component or socket.

The `setModifiability` operation triggers the execution of the 'set modifiability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.2.1.

The `min_number_of_modifications` parameter specifies the value of the 'min number of modifications' parameter of the 'set modifiability' action.

The `max_number_of_modifications` parameter specifies the value of the 'max number of modifications' parameter of the 'set modifiability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.21.61 `setModificationStatus` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setModificationStatus`	
Result:	`void`	
In:	`ModificationStatusValue`	`modification_state`
Exception:	`InvalidTarget`	

Description:

This operation assigns a value to the ’modification status’ of an rt-component or a socket.

The `setModificationStatus` operation triggers the execution of the ’set modification status’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.2.2.

The `modification_state` parameter specifies the value of the ’modification state’ parameter of the ’set modification status’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.62 `setModificationPresentationEffectResponsibility` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`setModificationPresentationEffectResponsibility`	
Result:	`void`	
In:	`Responsibility`	`the_responsibility`
Exception:	`InvalidTarget`	

Description:

This operation assigns a value to the ’modification presentation effect responsibility’ of an rt-component or a socket. This attribute indicates if it is the MHEG engine or the author who is responsible for reflecting a new state of the component or socket.

The `setModificationPresentationEffectResponsibility` operation triggers the execution of the ’set modification presentation effect responsibility’ elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.2.3.

The `the_responsibility` parameter specifies the value of the ’responsibility’ parameter of the ’set modification presentation effect responsibility’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.63 `getModifiability` operation

Synopsis:

Interface:	`RtComponentOrSocket`	
Operation:	`getModifiability`	
Result:	`void`	
Out:	`unsigned short`	`min_numbers_of_modifications`
Out:	`unsigned short`	`max_numbers_of_modifications`
Exception:	`InvalidTarget`	

Description:

This operation retrieves the ’minimum number of modifications required’ and the ’maximum number of modifications required’. If the ’maximum number of modifications required’ is equal to 0 the ’modifiability’ of the rt-component or socket is ’not modifiable’.

The `getModifiability` operation triggers the execution of the 'get modifiability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.1.

The `min_numbers_of_modifications` parameter specifies the value of the 'min numbers of modifications' parameter of the 'get modifiability' action.

The `max_numbers_of_modifications` parameter specifies the value of the 'max numbers of modifications' parameter of the 'get modifiability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.64 `getEffectiveModifiability` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getEffectiveModifiability`
Result:	`EffectiveModifiability`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'effective modifiability' attribute value of the rt-component or socket.

The `getEffectiveModifiability` operation triggers the execution of the 'get effective modifiability' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.65 `getModificationStatus` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getModificationStatus`
Result:	`ModificationStatusValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'modification status' value of the rt-component or socket.

The `getModificationStatus` operation triggers the execution of the 'get modification status' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.66 `getModificationMode` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getModificationMode`
Result:	`ModificationModeValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'modification mode' attribute value of the rt-component or socket.

The `getModificationMode` operation triggers the execution of the 'get modification mode' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.4.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.67 `getModificationPresentationEffectResponsibility` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`getModificationPresentationEffectResponsibility`
Result:	`Responsibility`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'modification presentation effect responsibility' attribute value of the rt-component or socket.

The `getModificationPresentationEffectResponsibility` operation triggers the execution of the 'get modification presentation effect responsibility' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.6.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.68 `setNoInteractionStyle` operation

Synopsis:

Interface:	`RtComponentOrSocket`
Operation:	`setNoInteractionStyle`
Result:	`void`
Exception:	`InvalidTarget`

Description:

This operation deassigns the currently assigned interaction style to an rt-component or a socket.

The `setNoInteractionStyle` operation triggers the execution of the 'set no style' elementary action with the bound rt-component or socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.6.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.21.69 IDL description

```
interface RtComponentOrSocket                               {

   void
      setRGS(
         in ChannelIdentifier
            channel_identifier)
   raises(InvalidTarget);

   RGSValue
      getRGS()
   raises(InvalidTarget);
```

```
void
    setOpacity(
        in unsigned short
            opacity_rate,
        in unsigned long
            transition_duration)
raises(InvalidTarget);

void
    setPresentationPriority(
        in PresentationPriority
            presentation_priority,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

unsigned short
    getOpacity()
raises(InvalidTarget);

unsigned short
    getEffectiveOpacity()
raises(InvalidTarget);

unsigned short
    getPresentationPriority()
raises(InvalidTarget);

void
    setVisibleDuration(
        in TemporalPosition
            initial_temporal_position,
        in TemporalPosition
            terminal_temporal_position)
raises(InvalidTarget, InvalidParameter);

void
    setTemporalTermination(
        in TemporalTermination
            temporal_termination)
raises(InvalidTarget);

void
    setCurrentTemporalPosition(
        in TemporalPosition
            temporal_position)
raises(InvalidTarget, InvalidParameter);

void
    setSpeed(
        in Speed
            the_speed,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setTimestones(
        in sequence<Timestone>
            timestones)
raises(InvalidTarget, InvalidParameter);

unsigned long
    getInitialTemporalPosition()
raises(InvalidTarget);

unsigned long
    getTerminalTemporalPosition()
raises(InvalidTarget);

unsigned long
    getVDLength(
        in GTIndicator
            gt_indicator)
raises(InvalidTarget);
```

```
TemporalTermination
   getTemporalTermination()
raises(InvalidTarget);

unsigned long
    getCurrentTemporalPosition()
raises(InvalidTarget);

short
    getSpeedRate()
raises(InvalidTarget);

unsigned long
    getOGTR()
raises(InvalidTarget);

short
    getEffectiveSpeedRate()
raises(InvalidTarget);

unsigned long
    getEffectiveOGTR()
raises(InvalidTarget);

unsigned short
    getTimestoneStatus()
raises(InvalidTarget);

void
   setPerceptibleSizeProjection(
       in PerceptibleSizeProjection
           perceptible_size_projection,
       in unsigned long
           transition_duration)
raises(InvalidTarget, InvalidParameter);

void
   setAspectRatio(
       in AspectRatio
           preserved)
raises(InvalidTarget);

void
   setVisibleSize(
        in VSGS
            the_vsgs,
        in VS
            the_vs,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
   setVisibleSizesAdjustment(
        in sequence<AdjustmentAxis>
            set_of_axes,
        in AdjustmentPolicy
            adjustment_policy,
        in unsigned long
            transition_duration)
raises(InvalidTarget);

void
   setBox(
       in BoxConstants
           box)
raises(InvalidTarget);

void
   setDefaultBackground(
       in unsigned short
           background,
       in unsigned long
           transition_duration)
raises(InvalidTarget, InvalidParameter);
```

```
void
   setAttachmentPoint(
       in AttachmentPointType
          type,
       in AttachmentPoint
          positions)
raises(InvalidTarget, InvalidParameter);

void
   setAttachmentPointPosition(
       in AttachmentPointType
          type,
       in ReferenceType
          vseap_reference_point,
       in Lengths
          the_lengths,
       in unsigned long
          transition_duration)
raises(InvalidTarget, InvalidParameter);

void
   setVisibleSizesAlignment(
        in SizeBorder
           size_border,
        in long
           interval,
        in unsigned long
           transition_duration)
raises(InvalidTarget);

void
   setMovingAbility(
       in UserControls
          moving_ability)
raises(InvalidTarget);

void
   setResizingAbility(
       in UserControls
          resizing_ability)
raises(InvalidTarget);

void
   setScalingAbility(
       in UserControls
          scaling_ability)
raises(InvalidTarget);

void
   setScrollingAbility(
       in UserControls
          scrolling_ability)
raises(InvalidTarget);

unsigned short
    getGSR()
raises(InvalidTarget);

SpecifiedPosition
   getPS(
       in GSIndicator
          gs)
raises(InvalidTarget);

AspectRatio
   getAspectRatio()
raises(InvalidTarget);

SpecifiedPosition
    getPSAP(
        in PointType
           point_type)
raises(InvalidTarget);
```

```
VSGS
   getVSGS()
raises(InvalidTarget);

SpecifiedPosition
   getVS()
raises(InvalidTarget);

BoxConstants
    getBox()
raises(InvalidTarget);

unsigned short
    getDefaultBackground()
raises(InvalidTarget);

SpecifiedPosition
    getVSIAP(
        in PointType
            point_type)
raises(InvalidTarget);

SpecifiedPosition
    getVSIAPPosition()
raises(InvalidTarget);

SpecifiedPosition
    getVSEAP(
        in PointType
            point_type)
raises(InvalidTarget);

SpecifiedPosition
    getVSEAPPosition(
        in ReferencePoint
            reference_point)
raises(InvalidTarget);

UserControls
    getMovingAbility()
raises(InvalidTarget);

UserControls
    getResizingAbility()
raises(InvalidTarget);

UserControls
   getScalingAbility()
raises(InvalidTarget);

UserControls
    getScrollingAbility()
raises(InvalidTarget);

void
   setSelectability(
       in unsigned short
           min_number_of_selections,
       in unsigned short
           max_number_of_selections)
raises(InvalidTarget, InvalidParameter);

void
   setSelectionStatus(
       in SelectionStatusValue
           selection_state)
raises(InvalidTarget);

void
   setSelectionPresentationEffectResponsibility(
       in Responsibility
           the_responsibility)
raises(InvalidTarget);
```

```
    void
       getSelectability(
           out unsigned short
               min_number_of_selections,
           out unsigned short
               max_number_of_selections)
    raises(InvalidTarget);

    EffectiveSelectability
        getEffectiveSelectability()
    raises(InvalidTarget);

    SelectionStatusValue
        getSelectionStatus()
    raises(InvalidTarget);

    SelectionModeValue
        getSelectionMode()
    raises(InvalidTarget);

    Responsibility
       getSelectionPresentationEffectResponsibility()
    raises(InvalidTarget);

    void
       setModifiability(
           in unsigned short
               min_number_of_modifications,
           in unsigned short
               max_number_of_modifications)
    raises(InvalidTarget, InvalidParameter);

    void
       setModificationStatus(
           in ModificationStatusValue
               modification_state)
    raises(InvalidTarget);

    void
       setModificationPresentationEffectResponsibility(
           in Responsibility
               the_responsibility)
    raises(InvalidTarget);

    void
       getModifiability(
           out unsigned short
               min_numbers_of_modifications,
           out unsigned short
               max_numbers_of_modifications)
    raises(InvalidTarget);

    EffectiveModifiability
        getEffectiveModifiability()
    raises(InvalidTarget);

    ModificationStatusValue
        getModificationStatus()
    raises(InvalidTarget);

    ModificationModeValue
        getModificationMode()
    raises(InvalidTarget);

    Responsibility
        getModificationPresentationEffectResponsibility()
    raises(InvalidTarget);

    void
       setNoInteractionStyle()
    raises(InvalidTarget);

};
```

7.1.22 `RtComponent` object

For the `RtComponent` object no specific operations are defined. The object inherits from the `RtComponentOrSocket` object and from the `RtObject` object.

7.1.22.1 IDL description

```
interface RtComponent: RtComponentOrSocket, RtObject {};
```

7.1.23 `RtCompositeOrStructuralSocket` object

The following subclause defines the operations of the `RtCompositeOrStructuralSocket` object.

7.1.23.1 `setResizingStrategy` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`	
Operation:	`setResizingStrategy`	
Result:	`void`	
In:	`ResizingStrategy`	`resizing_strategy`
Exception:	`InvalidTarget`	

Description:

This operation specifies the PS resizing strategy that an rt-composite or structural socket is to have regarding the modification of the VSs of the child sockets having a Parent Relative Generic Space (PRGS).

The `setResizingStrategy` operation triggers the execution of the ’set resizing strategy’ elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.2.2.

The `resizing_strategy` parameter specifies the value of the ’resizing strategy’ parameter of the ’set resizing strategy’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.2 `getResizingStrategy` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`
Operation:	`getResizingStrategy`
Result:	`ResizingStrategy`
Exception:	`InvalidTarget`

Description:

This operation retrieves the resizing strategy value of the rt-composite or structural socket.

The `getResizingStrategy` operation triggers the execution of the ’get resizing strategy’ elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 53.4.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.3 `setAudibleCompositionEffect` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`	
Operation:	`setAudibleCompositionEffect`	
Result:	`void`	
In:	`unsigned short`	`audible_effect`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	

Description:

This operation specifies the audible composition effect of an rt-composite or a structural socket. This effect is to be propagated to their descendant sockets having a PRGS. It is used to calculate the effective Original Volume (OV) of the descendant sockets having a PRGS.

The `setAudibleCompositionEffect` operation triggers the execution of the ’set audible composition effect’ elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.2.2.

The `audible_effect` parameter specifies the value of the ’audible effect’ parameter of the ’set audible composition effect’ action.

The `transition_duration` parameter specifies the value of the ’transition duration’ parameter of the ’set audible composition effect’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.4 `getAudibleCompositionEffect` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`
Operation:	`getAudibleCompositionEffect`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the audible composition effect value of the rt-composite or structural socket. This effect is expressed as a percentage and used to determine the effective OV of the child sockets of the rt-composite or structural socket having as PRGS the rt-composite or structural socket. This effect is recursive for the child sockets of the structural sockets having as PRGS the rt-composite or structural socket, and so on.

The `getAudibleCompositionEffect` operation triggers the execution of the ’get audible composition effect’ elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.3.3.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.5 `getNumberOfSelectedSockets` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`
Operation:	`getNumberOfSelectedSockets`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the ’number of selected sockets’ attribute value of the rt-composite or structural socket.

The `getNumberOfSelectedSockets` operation triggers the execution of the 'get number of selected sockets' elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 57.3.5.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.6 `getNumberOfModifiedSockets` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`
Operation:	`getNumberOfModifiedSockets`
Result:	`unsigned short`
Exception:	`InvalidTarget`

Description:

This operation retrieves the 'number of modified sockets' attribute value of the rt-composite or structural socket.

The `getNumberOfModifiedSockets` operation triggers the execution of the 'get number of modified sockets' elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 58.3.5.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.23.7 `setMenuInteractionStyle` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`	
Operation:	`setMenuInteractionStyle`	
Result:	`void`	
In:	`Orientation`	`upper_menu_orientation`
In:	`sequence <Association>`	`list_of_associations`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns the menu interaction style to an rt-composite or a structural socket. This operation defines a style which affects the complete rt-composite or structural socket, i.e all generations.

The `setMenuInteractionStyle` operation triggers the execution of the 'set menu style' elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.4.

The `upper_menu_orientation` parameter specifies the value of the 'upper menu orientation' parameter of the 'set menu style' action.

The `list_of_associations` parameter specifies the value of the 'list of associations' parameter of the 'set menu style' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.23.8 `setScrollingListInteractionStyle` operation

Synopsis:

Interface:	`RtCompositeOrStructuralSocket`	
Operation:	`setScrollingListInteractionStyle`	
Result:	`void`	
In:	`PerceptibleReference`	`background`
In:	`unsigned short`	`visible_items_number`
In:	`SocketTail`	`first_item`
In:	`Separator`	`the_separator`
In:	`Orientation`	`the_orientation`
In:	`SliderSide`	`slider_side`
In:	`PerceptibleReference`	`slider`
In:	`PerceptibleReference`	`slider_cursor`
In:	`PerceptibleReference`	`slider_background`
In:	`long`	`slider_min_value`
In:	`long`	`slider_max_value`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns the scrolling list interaction style to an rt-composite or a structural socket. This operation defines a style which affects the first generation and only the child presentable sockets of the rt-composite or structural socket.

The `setScrollingListInteractionStyle` operation triggers the execution of the 'set scrolling list style' elementary action with the bound rt-composite or structural socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.5.

The `background` parameter specifies the value of the 'background' parameter of the 'set scrolling list style' action.

The `visible_items_number` parameter specifies the value of the 'number of visible items' parameter of the 'set scrolling list style' action.

The `first_item` parameter specifies the value of the 'first item' parameter of the 'set scrolling list style' action.

The `the_separator` parameter specifies the value of the 'separator' parameter of the 'set scrolling list style' action.

The `the_orientation` parameter specifies the value of the 'scrolling list orientation' parameter of the 'set scrolling list style' action.

The `slider_side` parameter specifies the value of the 'slider side' parameter of the 'set scrolling list style' action.

The `slider, slider_cursor, slider_background, slider_min_value, slider_max_value` parameters specify the values of the parameters of the 'set slider style' action embedded by the 'set scrolling list style' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.23.9 IDL description

```
interface RtCompositeOrStructuralSocket                     {

    void
        setResizingStrategy(
            in ResizingStrategy
                resizing_strategy)
    raises(InvalidTarget);

    ResizingStrategy
         getResizingStrategy()
    raises(InvalidTarget);

    void
        setAudibleCompositionEffect(
            in unsigned short
                audible_effect,
            in unsigned long
                transition_duration)
    raises(InvalidTarget);

    unsigned short
         getAudibleCompositionEffect()
    raises(InvalidTarget);

    unsigned short
        getNumberOfSelectedSockets()
    raises(InvalidTarget);

    unsigned short
         getNumberOfModifiedSockets()
    raises(InvalidTarget);

    void
        setMenuInteractionStyle(
            in Orientation
                upper_menu_orientation,
            in sequence <Association>
                list_of_associations)
    raises(InvalidTarget, InvalidParameter);

    void
        setScrollingListInteractionStyle(
            in PerceptibleReference
                background,
            in unsigned short
                visible_items_number,
            in SocketTail
                first_item,
            in Separator
                the_separator,
            in Orientation
                the_orientation,
            in SliderSide
                slider_side,
            in PerceptibleReference
                slider,
            in PerceptibleReference
                slider_cursor,
            in PerceptibleReference
                slider_background,
            in long
                slider_min_value,
            in long
                slider_max_value)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.24 `RtComposite` object

For the `RtComposite` object no specific operations are defined. The object inherits from the `RtCompositeOrStructuralSocket` object and from the `RtComponent` object.

7.1.24.1 IDL description

```
interface RtComposite: RtCompositeOrStructuralSocket, RtComponent  {};
```

7.1.25 `StructuralSocket` object

For the `StructuralSocket` object no specific operations are defined. The object inherits from the `RtCompositeOrStructuralSocket` object and from the `Socket` object.

7.1.25.1 IDL description

```
interface StructuralSocket: RtCompositeOrStructuralSocket, Socket {};
```

7.1.26 `RtGenericContentOrPresentableSocket` object

The following subclause defines the operations of the `RtGenericContentOrPresentableSocket` object.

7.1.26.1 `setAudibleVolume` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`	
Operation:	`setAudibleVolume`	
Result:	`void`	
In:	`AudibleVolume`	`audible_volume`
In:	`unsigned long`	`transition_duration`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This action specifies the audible volume of an rt-content, an rt-multiplexed content, a presentable socket or a multiplexed presentable socket.

The `setAudibleVolume` operation triggers the execution of the ’set audible volume’ elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.2.1.

The `audible_volume` parameter specifies the value of the ’audible volume’ parameter of the ’set audible volume’ action.

The `transition_duration` parameter specifies the value of the ’transition duration’ parameter of the ’set audible volume’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.26.2 `getInitialOriginalAudibleVolume` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`
Operation:	`getInitialOriginalAudibleVolume`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the initial original audible volume value of the rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket. This initial volume is expressed in original generic audible volume unit within the interval defined by the original audible volume range.

The `getInitialOriginalAudibleVolume` operation triggers the execution of the ’get IOV’ elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.26.3 `getCurrentOriginalAudibleVolume` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`
Operation:	`getCurrentOriginalAudibleVolume`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the current original audible volume value of the rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket. This current volume is expressed in original generic audible volume unit within the interval defined by the original audible volume range.

The `getCurrentOriginalAudibleVolume` operation triggers the execution of the ’get current OV’ elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.3.2.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.26.4 `getEffectiveOriginalAudibleVolume` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`
Operation:	`getEffectiveOriginalAudibleVolume`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the effective original audible volume value of the rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket. This effective volume is expressed in original generic audible volume unit within the interval defined by the original audible volume range. It is calculated by the MHEG engine using the current original audible volume and the audible composition effect.

The `getEffectiveOriginalAudibleVolume` operation triggers the execution of the ’get effective OV’ elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.3.4.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.26.5 `getPerceptibleAudibleVolume` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`
Operation:	`getPerceptibleAudibleVolume`
Result:	`unsigned long`
Exception:	`InvalidTarget`

Description:

This operation retrieves the perceptible original audible volume value of the rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket in the assigned channel. This perceptible volume is expressed in channel generic audible volume unit within the interval defined by the channel audible volume range. It is calculated by the MHEG engine and corresponds to a projection of the effective original audible volume in the channel generic space.

The `getPerceptibleAudibleVolume` operation triggers the execution of the 'get perceptible OV' elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 54.3.5.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.26.6 `setButtonInteractionStyle` operation

Synopsis:

Interface:	`RtGenericContentOrPresentableSocket`	
Operation:	`setButtonInteractionStyle`	
Result:	`void`	
In:	`PresentationState`	`initial_state`
In:	`AlternatePresentation`	`alternate_presentation_1`
In:	`AlternatePresentation`	`alternate_presentation_2`
In:	`AlternatePresentation`	`alternate_presentation_3`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns the button interaction style to an rt-content, rt-multiplexed content, a presentable socket or a multiplexed presentable socket.

The `setButtonInteractionStyle` operation triggers the execution of the 'set button style' elementary action with the bound rt-content, rt-multiplexed content, presentable socket or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.1.

The `initial_state` parameter specifies the value of the 'initial state' parameter of the 'set button style' action.

The `alternate_presentation_1` parameter specifies the value of the 'alternate presentation 1' parameter of the 'set button style' action.

The `alternate_presentation_2` parameter specifies the value of the 'alternate presentation 2' parameter of the 'set button style' action.

The `alternate_presentation_3` parameter specifies the value of the 'alternate presentation 3' parameter of the 'set button interaction style' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.26.7 IDL description

```
interface RtGenericContentOrPresentableSocket               {

    void
        setAudibleVolume(
            in AudibleVolume
                audible_volume,
            in unsigned long
                transition_duration)
    raises(InvalidTarget, InvalidParameter);

    unsigned long
        getInitialOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getCurrentOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getEffectiveOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getPerceptibleAudibleVolume()
    raises(InvalidTarget);

    void
        setButtonInteractionStyle(
            in PresentationState
                initial_state,
            in AlternatePresentation
                alternate_presentation_1,
            in AlternatePresentation
                alternate_presentation_2,
            in AlternatePresentation
                alternate_presentation_3)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.27 `RtGenericContent` object

For the `RtGenericContent` object no specific operations are defined. The object inherits from the `RtGenericContentOrPresentableSocket` object and from the `RtComponent` object.

7.1.27.1 IDL description

```
interface RtGenericContent: RtGenericContentOrPresentableSocket, RtComponent {};
```

7.1.28 `GenericPresentableSocket` object

For the `GenericPresentableSocket` object no specific operations are defined. The object inherits from the `RtGenericContentOrPresentableSocket` object and from the `Socket` object.

7.1.28.1 IDL description

```
interface GenericPresentableSocket: RtGenericContentOrPresentableSocket, Socket
{};
```

7.1.29 `RtContentOrPresentableSocket` object

The following subclause defines the operations of the `RtContentOrPresentableSocket` object.

7.1.29.1 `setSliderInteractionStyle` operation

Synopsis:

Interface:	`RtContentOrPresentableSocket`	
Operation:	`setSliderInteractionStyle`	
Result:	`void`	
In:	`PerceptibleReference`	`cursor`
In:	`PerceptibleReference`	`background`
In:	`Orientation`	`the_orientation`
In:	`short`	`min_value`
In:	`short`	`max_value`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns the slider interaction style to an rt-content or a presentable socket created from a content object model which contains a generic numeric as ’content data’.

The `setSliderStyle` operation triggers the execution of the ’set slider style’ elementary action with the bound rt-content or presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.2.

The `cursor` parameter specifies the value of the ’cursor’ parameter of the ’set slider style’ action.

The `background` parameter specifies the value of the ’background’ parameter of the ’set slider style’ action.

The `the_orientation` parameter specifies the value of the ’orientation’ parameter of the ’set slider style’ action.

The `min_value` parameter specifies the value of the ’minimum value’ parameter of the ’set slider style’ action.

The `max_value` parameter specifies the value of the ’maximum value’ parameter of the ’set slider style’ action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.29.2 `setEntryFieldInteractionStyle` operation

Synopsis:

Interface:	`RtContentOrPresentableSocket`	
Operation:	`setEntryFieldInteractionStyle`	
Result:	`void`	
In:	`EchoStyle`	`echo_style`
In:	`PerceptibleReference`	`background`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation assigns the entry field interaction style to an rt-content or a presentable socket created from a content object model which contains a generic numeric or a generic string as ’content data’.

The `setEntryFieldInteractionStyle` operation triggers the execution of the ’set entry field style’ elementary action with the bound rt-content or presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 59.2.3.

The echo_style parameter specifies the value of the ’echo style’ parameter of the ’set entry field style’ action.

The background parameter specifies the value of the ’background’ parameter of the ’set entry field style’ action.

The InvalidTarget exception is raised when the object instance does not represent a valid target for the normal completion of the action. The period member returns the current period of the target.

The InvalidParameter exception is raised when the value of one of the parameters prohibits the normal execution of the action. The completion_status member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The parameter_number member identifies the rank of the invalid parameter.

7.1.29.3 IDL description

```
interface RtContentOrPresentableSocket                    {

    void
        setSliderInteractionStyle(
            in PerceptibleReference
                cursor,
            in PerceptibleReference
                background,
            in Orientation
                the_orientation,
            in short
                min_value,
            in short
                max_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setEntryFieldInteractionStyle(
            in EchoStyle
                echo_style,
            in PerceptibleReference
                background)
    raises(InvalidTarget, InvalidParameter);

};
```

7.1.30 RtContent object

For the RtContent object no specific operations are defined. The object inherits from the RtContentOrPresentableSocket object and from the RtGenericContent object.

7.1.30.1 IDL description

```
interface RtContent: RtContentOrPresentableSocket, RtGenericContent {};
```

7.1.31 PresentableSocket object

For the PresentableSocket object no specific operations are defined. The object inherits from the RtContentOrPresentableSocket object and from the GenericPresentableSocket object.

7.1.31.1 IDL description

```
interface PresentableSocket: RtContentOrPresentableSocket,
GenericPresentableSocket {};
```

7.1.32 RtMultiplexedContentOrPresentableSocket object

The following subclause defines the operations of the RtMultiplexedContentOrPresentableSocket object.

7.1.32.1 `setStreamChoice` operation

Synopsis:

Interface:	`RtMultiplexedContentOrPresentableSocket`	
Operation:	`setStreamChoice`	
Result:	`void`	
In:	`StreamIdentifier`	`stream_identifier`
Exception:	`InvalidTarget`	
Exception:	`InvalidParameter`	

Description:

This operation specifies a stream to be chosen in the multiplexed data and assigned to the rt-multiplexed content or multiplexed presentable socket. Once a stream is chosen for an rt-multiplexed content or a multiplexed presentable socket, when it becomes running, the rt-multiplexed content or the multiplexed presentable socket is responsible for the presentation of this chosen stream.

The `setStreamChoice` operation triggers the execution of the ’set stream choice’ elementary action with the bound rt-multiplexed content or multiplexed presentable socket as its single target.

The `stream_identifier` parameter specifies the value of the ’stream choice’ parameter of the ’set stream choice’ action.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 55.2.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.32.2 `getStreamChosen` operation

Synopsis:

Interface:	`RtMultiplexedContentOrPresentableSocket`
Operation:	`getStreamChosen`
Result:	`StreamValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the stream chosen for the rt-multiplexed content or multiplexed presentable socket.

The `getStreamChosen` operation triggers the execution of the ’get stream chosen’ elementary action with the bound rt-multiplexed content or multiplexed presentable socket as its single target.

The effect of the action on its target, the semantics of its parameters, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 55.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.32.3 IDL description

```
interface RtMultiplexedContentOrPresentableSocket           {

    void
        setStreamChoice(
            in StreamIdentifier
                stream_identifier)
    raises(InvalidTarget, InvalidParameter);

    StreamValue
        getStreamChosen()
    raises(InvalidTarget);

};
```

7.1.33 `RtMultiplexedContent` object

For the `RtMultiplexedContent` object no specific operations are defined. The object inherits from the `RtMultiplexedContentOrPresentableSocket` object and from the `RtGenericContent` object.

7.1.33.1 IDL description

```
interface RtMultiplexedContent: RtMultiplexedContentOrPresentableSocket,
RtGenericContent {};
```

7.1.34 `MultiplexedPresentableSocket` object

For the `MultiplexedPresentableSocket` object no specific operations are defined. The object inherits from the `RtMultiplexedContentOrPresentableSocket` object and from the `GenericPresentableSocket` object.

7.1.34.1 IDL description

```
interface MultiplexedPresentableSocket: RtMultiplexedContentOrPresentableSocket,
GenericPresentableSocket {};
```

7.1.35 `Channel` object

The following subclause defines the operations of the `Channel` object. The object inherits from the `Entity` object.

7.1.35.1 `bind` operation

Synopsis:

Interface:	`Channel`	
Operation:	`bind`	
Result:	`ChannelIdentifier`	
In:	`ChannelReference`	`channel_reference`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation binds the Channel instance (an interface object instance) with a channel (an MHEG entity).

The `channel_reference` parameter specifies the reference of the channel.

The operation returns the identifier of the bound channel.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.35.2 `unbind` operation

Synopsis:

Interface:	`Channel`
Operation:	`unbind`
Result:	`void`
Exception:	`NotBound`

Description:

This operation cancels the binding between the Channel instance (an interface object instance) and a channel (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.35.3 `new` operation

Synopsis:

Interface:	`Channel`	
Operation:	`new`	
Result:	`ChannelIdentifier`	
In:	`ChannelReference`	`channel_reference`
In:	`OriginalDefDeclaration`	`original_definition_declaration`
Exception:	`AlreadyBound`	
Exception:	`InvalidTarget`	

Description:

This operation enables the creation of a channel by the MHEG engine.

The `new` operation triggers the execution of the 'new channel' elementary action targeted at a single channel.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 61.2.1.

The `channel_reference` parameter specifies a reference to a channel.

The `original_definition_declaration` parameter specifies the value of the 'original definition declaration' parameter of the 'new channel' action.

This operation implicitly binds the Channel instance (an interface object instance) with the new created channel (an MHEG entity).

The operation returns the identifier of the new created channel bound with the Channel instance.

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.35.4 `delete` operation

Synopsis:

Interface:	`Channel`
Operation:	`delete`
Result:	`void`
Exception:	`NotBound`
Exception:	`InvalidTarget`

Description:

This operation enables the removing of a channel by the MHEG engine.

The `delete` operation triggers the execution of the 'delete channel' elementary action targeted at a single channel.

The effect of the action on its target and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 61.2.2.

This operation implicitly cancels the binding between the Channel instance (an interface object instance) and the new deleted channel (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.35.5 `getRtAvailabilityStatus` operation

Synopsis:

Interface:	`Channel`
Operation:	`getAvailability`
Result:	`ChannelStatusValue`
Exception:	`NotBound`
Exception:	`InvalidTarget`

Description:

This operation retrieves the availability of a channel to the MHEG engine.

The `getAvailability` operation triggers the execution of the 'get channel availability status' elementary action with the bound channel as its single target.

The effect of the action on its target, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 61.3.1.

The operation returns the availability of the channel bound with the Channel instance. The returned value is either `NOT_AVAILABLE`, `PROCESSING` or `AVAILABLE`.

When the returned value is `NOT_AVAILABLE`, the operation implicitly cancels the binding between the Channel instance (an interface object instance) and the channel (an MHEG entity).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.35.6 `getIdentifier` operation

Synopsis:

Interface:	`Channel`
Operation:	`getIdentifier`
Result:	`ChannelIdentifier`
Exception:	`NotBound`

Description:

This operation retrieves the identifier of the channel (an MHEG entity) bound with the Channel instance (an interface object instance).

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.35.7 `kill` operation

Synopsis:

Interface:	`Channel`
Operation:	`kill`
Result:	`void`

Description:

This operation deletes the Channel instance (an interface object instance).

7.1.35.8 `setPerceptability` operation

Synopsis:

Interface:	`Channel`	
Operation:	`setPerceptability`	
Result:	`void`	
In:	`ChannelPerceptabilityValue`	`channel_perceptability`
Exception:	`InvalidTarget`	

Description:

This operation enables to turn on or off a channel. This is used to enable or disable the perception of a channel by a user.

The `setPerceptability` operation triggers the execution of the 'set channel perceptability' elementary action with the bound channel as its single target.

The effect of the action on its target, the semantics of its parameters and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 62.2.1.

The `channel_perceptability` parameter specifies the value of the 'perceptability' parameter of the 'set channel perceptability' action.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.35.9 `getPerceptability` operation

Synopsis:

Interface:	`Channel`
Operation:	`getPerceptability`
Result:	`ChannelPerceptabilityValue`
Exception:	`InvalidTarget`

Description:

This operation retrieves the perceptability of a channel.

The `getPerceptability` operation triggers the execution of the 'get channel perceptability' elementary action with the bound channel as its single target.

The effect of the action on its target, the computation of its result and the error conditions that cause exceptions to be raised are defined by the MHEG standard, 62.3.1.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.35.10 `getAssignedPerceptibles` operation

Synopsis:

Interface:	`Channel`
Operation:	`getAssignedPerceptibles`
Result:	`sequence<PerceptibleReference>`
Exception:	`InvalidTarget`

Description:

This operation retrieves the perceptibles assigned to the channel.

The `getAssignedPerceptibles` operation has no corresponding MHEG elementary action. It is symmetrical to the 'get RGS' elementary action which retrieves the channel assigned to a perceptible.

The `InvalidTarget` exception is raised when the object instance does not represent a valid target for the normal completion of the action. The `period` member returns the current period of the target.

7.1.35.11 IDL description

```
interface Channel: Entity                                    {

   ChannelIdentifier
       bind(
           in ChannelReference
               channel_reference)
   raises(AlreadyBound, InvalidTarget);

   void
      unbind()
   raises(NotBound);
```

```
    ChannelIdentifier
        new(
            in ChannelReference
                channel_reference,
            in OriginalDefDeclaration
                original_definition_declaration)
    raises(AlreadyBound, InvalidTarget);

    void
       delete()
    raises(NotBound, InvalidTarget);

    ChannelStatusValue
        getAvailability()
    raises(NotBound, InvalidTarget);

    ChannelIdentifier
        getIdentifier()
    raises(NotBound);

    void
       kill();

    void
       setPerceptability(
           in ChannelPerceptabilityValue
               channel_perceptability)
    raises(InvalidTarget);

    ChannelPerceptabilityValue
        getPerceptability()
    raises(InvalidTarget);

    sequence<PerceptibleReference>
        getAssignedPerceptibles()
    raises(InvalidTarget);

};
```

7.1.36 Parameter definition

The following subclause defines the parameters that are used by the mandatory primitives.

```
//==========================================================================
typedef sequence<long> ApplicationIdentifier;

// Corresponding MHEG datatype: Object-Number
//==========================================================================
typedef long ObjectNumber;

// Interface: MhObject Operation: bind
// Interface: MhObject Operation: prepare
// Interface: MhObject Operation: getIdentifier
// Corresponding MHEG datatype: MHEG-Identifier
//==========================================================================
struct MHEGIdentifier {
    sequence<ApplicationIdentifier,1>
        application_identifier;
    ObjectNumber
        object_number;
};

// Corresponding MHEG datatype: Public-Identifier
//==========================================================================
typedef string PublicIdentifier;

// Corresponding MHEG datatype: System-Identifier
//==========================================================================
typedef string SystemIdentifier;
```

```
// Corresponding MHEG datatype: External-Long-Identifier
//==========================================================================
struct ExternalLongIdentifier {
    PublicIdentifier
        public_identifier;
    SystemIdentifier
        system_identifier;
};

// Corresponding MHEG datatype: Alias
//==========================================================================
typedef string Alias;

// Corresponding MHEG datatype: Container-Child-Reference
//==========================================================================
enum ContainerChildReference {
    CHILD,
    DESCENDANT
};

// Interface: MhObject Operation: getPreparationStatus
// Corresponding MHEG datatype: Preparation-Status-Value
//==========================================================================
enum PreparationStatusValue {
    READY,
    NOT_READY,
    PROCESSING
};

// Interface: MhMultiplexedContent Operation: setMultiplex
// Interface: MhMultiplexedContent Operation: setDemultiplex
// Interface: RtMultiplexedContentOrPresentableSocket Operation: setStreamChoice
// Corresponding MHEG datatype: Stream-Identifier
//==========================================================================
typedef sequence<long> StreamIdentifier;

// Corresponding MHEG datatype: Rt-Dynamic-Reference
//==========================================================================
enum RtDynamicReference {
    QUESTION_MARK,
    STAR
};

// Interface: RtObject Operation: getAvailabilityStatus
// Corresponding MHEG datatype: Rt-Availibility-Status-Value
//==========================================================================
enum RtAvailabilityStatusValue {
    RT_AVAILIBILITY_STATUS_VALUE_AVAILABLE,
    RT_AVAILIBILITY_STATUS_VALUE_NOT_AVAILABLE,
    RT_AVAILIBILITY_STATUS_VALUE_PROCESSING
};

// Interface: RtObject Operation: getRunningStatus
// Corresponding MHEG datatype: Running-Status-Value
//==========================================================================
enum RunningStatusValue {
    RUNNING_STATUS_VALUE_RUNNING,
    RUNNING_STATUS_VALUE_NOT_RUNNING,
    RUNNING_STATUS_VALUE_PROCESSING
};

// Interface: RtScript Operation: getTerminationStatus
// Corresponding MHEG datatype: Termination-Status-Value
//==========================================================================
enum TerminationStatusValue {
    TERMINATED,
    NOT_TERMINATED
};

// Interface: RtComponentOrSocket Operation: setRGS
// Interface: Channel Operation: getIdentifier
// Corresponding MHEG datatype: Channel-Identifier
//==========================================================================
typedef long ChannelIdentifier;
```

```
// Corresponding MHEG datatype: Priority-Level
//=========================================================================
enum PriorityLevel {
    INCREMENT_PRIORITY,
    DECREMENT_PRIORITY
};

// Interface: RtComponentOrSocket Operation: setVisibleDuration
// Corresponding MHEG datatype: Temporal-Position
//=========================================================================
enum TemporalPositionTag { SPECIFIED_TEMPORAL_POINT_TAG,
LOGICAL_TEMPORAL_PD_POINT_TAG };
union TemporalPosition
switch (TemporalPositionTag){
    case SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
            logical_temporal_PD_point;
};

// Interface: RtComponentOrSocket Operation: setCurrentTemporalPosition
// Corresponding MHEG datatype: Current-Temporal-Position
//=========================================================================
enum CurrentTemporalPositionTag {
CURRENT_TEMPORAL_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
CURRENT_TEMPORAL_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union CurrentTemporalPosition
switch (CurrentTemporalPositionTag){
    case CURRENT_TEMPORAL_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case CURRENT_TEMPORAL_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
            logical_temporal_vd_point;
};

// Interface: RtComponentOrSocket Operation: setTemporalTermination
// Interface: RtComponentOrSocket Operation: getTemporalTermination
// Corresponding MHEG datatype: Temporal-Termination
//=========================================================================
enum TemporalTermination {
    TEMPORAL_TERMINATION_FREEZE,
    TEMPORAL_TERMINATION_STOP
};

// Interface: RtComponentOrSocket Operation: setSpeed
// Corresponding MHEG datatype: Speed
//=========================================================================
enum SpeedTag { SPECIFIED_OGTR_TAG, SPEED_RATE_TAG, SCALING_FACTOR_TAG };
union Speed
switch (SpeedTag){
    case SPECIFIED_OGTR_TAG:
        long
            specified_OGTR;
    case SPEED_RATE_TAG:
        long
            speed_rate;
    case SCALING_FACTOR_TAG:
        long
            scaling_factor;
};

// Corresponding MHEG datatype: Timestone-Position
//=========================================================================
enum TimestonePositionTag { TIMESTONE_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
TIMESTONE_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG,
TIMESTONE_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union TimestonePosition
switch (TimestonePositionTag){
    case TIMESTONE_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case TIMESTONE_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
            logical_temporal_PD_point;
    case TIMESTONE_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
            logical_temporal_VD_point;
};

// Interface: RtComponentOrSocket Operation: getVDLength
// Corresponding MHEG datatype: GT-Indicator
//==========================================================================
enum GTIndicator {
    OGTU,
    RGTU
};

// Corresponding MHEG datatype: Perceptible-Projection
//==========================================================================
enum PerceptibleProjectionTag { SPECIFIED_SIZE_TAG, IOGSR_SCALING_FACTOR_TAG,
COGSR_SCALING_FACTOR_TAG };
union PerceptibleProjection
switch (PerceptibleProjectionTag){
    case SPECIFIED_SIZE_TAG:
        long
            specified_size;
    case IOGSR_SCALING_FACTOR_TAG:
        long
            iogsr_scaling_factor;
    case COGSR_SCALING_FACTOR_TAG:
        long
            cogsr_scaling_factor;
};

// Interface: RtComponentOrSocket Operation: setAspectRatioPreserved
// Interface: RtComponentOrSocket Operation: getAspectRatio
// Corresponding MHEG datatype: Aspect-Ratio
//==========================================================================
enum AspectRatio {
    PRESERVED,
    NOT_PRESERVED
};

// Interface: RtComponentOrSocket Operation: setVisibleSize
// Interface: RtComponentOrSocket Operation: getVSGS
// Corresponding MHEG datatype: VSGS
//==========================================================================
enum VSGS {
    THIS,
    RELATIVE
};

// Corresponding MHEG datatype: Size-Attribute
//==========================================================================
enum SizeAttributeTag { SIZE_ATTRIBUTE_SPECIFIED_SIZE_TAG,
SIZE_ATTRIBUTE_IVS_RELATIVE_TAG, SIZE_ATTRIBUTE_CVS_RELATIVE_TAG };
union SizeAttribute
switch (SizeAttributeTag){
    case SIZE_ATTRIBUTE_SPECIFIED_SIZE_TAG:
        long
            specified_size;
    case SIZE_ATTRIBUTE_IVS_RELATIVE_TAG:
        long
            ivs_relative;
    case SIZE_ATTRIBUTE_CVS_RELATIVE_TAG:
        long
            cvs_relative;
};
```

```
// Interface: RtComponentOrSocket Operation: setVisibleSizesAdjustment
// Corresponding MHEG datatype: Adjustment-Axis
//==========================================================================
enum AdjustmentAxis {
    X_AXIS,
    Y_AXIS,
    Z_AXIS
};

// Corresponding MHEG datatype: Sub-Socket-Reference
//==========================================================================
enum SubSocketReference {
    SUB_SOCKET_REFERENCE_CHILD,
    SUB_SOCKET_REFERENCE_DESCENDANT,
    SUB_SOCKET_REFERENCE_QUESTION_MARK_CHILD,
    SUB_SOCKET_REFERENCE_QUESTION_MARK_DESCENDANT
};

// Interface: RtComponentOrSocket Operation: setBox
// Interface: RtComponentOrSocket Operation: getBox
// Corresponding MHEG datatype: Box-Constants
//==========================================================================
enum BoxConstants {
    PRESENTED,
    NOT_PRESENTED
};

// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Reference-Type
//==========================================================================
enum ReferenceType {
    VSIAP,
    VSEAP
};

// Interface: RtComponentOrSocket Operation: setAttachmentPoint
// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Attachment-Point-Type
//==========================================================================
enum AttachmentPointType {
    ATTACHMENT_POINT_TYPE_PSAP,
    ATTACHMENT_POINT_TYPE_VSIAP,
    ATTACHMENT_POINT_TYPE_VSEAP
};

// Interface: RtComponentOrSocket Operation: setVisibleSizesAlignment
// Corresponding MHEG datatype: Size-Border
//==========================================================================
enum SizeBorder {
    TOP,
    BOTTOM,
    RIGHT,
    LEFT,
    UPPER_Z,
    LOWER_Z,
    CENTER_X,
    CENTER_Y,
    CENTER_Z
};

// Interface: RtComponentOrSocket Operation: setMovingAbility
// Interface: RtComponentOrSocket Operation: setResizingAbility
// Interface: RtComponentOrSocket Operation: setScalingAbility
// Interface: RtComponentOrSocket Operation: setScrollingAbility
// Interface: RtComponentOrSocket Operation: getMovingAbility
// Interface: RtComponentOrSocket Operation: getResizingAbility
// Interface: RtComponentOrSocket Operation: getScalingAbility
// Interface: RtComponentOrSocket Operation: getScrollingAbility
// Corresponding MHEG datatype: User-Controls
//==========================================================================
enum UserControls {
    ALLOWED,
    NOT_ALLOWED
};
```

```
// Interface: RtComponentOrSocket Operation: getPS
// Corresponding MHEG datatype: GS-Indicator
//==========================================================================
enum GSIndicator {
    OGSU,
    RGSU
};

// Interface: RtComponentOrSocket Operation: getPSAP
// Interface: RtComponentOrSocket Operation: getVSIAP
// Corresponding MHEG datatype: Point-Type
//==========================================================================
enum PointType {
    RELATIVE_POINT,
    ABSOLUTE_POINT
};

// Interface: RtComponentOrSocket Operation: setSelectionStatus
// Interface: RtComponentOrSocket Operation: getSelectionStatus
// Corresponding MHEG datatype: Selection-Status-Value
//==========================================================================
enum SelectionStatusValue {
    SELECTED,
    NOT_SELECTED
};

// Interface: RtComponentOrSocket Operation:
//   setSelectionPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   getSelectionPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   setModificationPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   getModificationPresentationEffectResponsibility
// Corresponding MHEG datatype: Responsibility
//==========================================================================
enum Responsibility {
    MHEG_ENGINE,
    AUTHOR
};

// Interface: RtComponentOrSocket Operation: getEffectiveSelectability
// Corresponding MHEG datatype: Effective-Selectability
//==========================================================================
enum EffectiveSelectability {
    EFFECTIVELY_SELECTABLE,
    EFFECTIVELY_NOT_SELECTABLE
};

// Interface: RtComponentOrSocket Operation: setModificationStatus
// Interface: RtComponentOrSocket Operation: getModificationStatus
// Corresponding MHEG datatype: Modification-Status-Value
//==========================================================================
enum ModificationStatusValue {
    MODIFIED,
    MODIFYING,
    NOT_MODIFIED
};

// Interface: RtComponentOrSocket Operation: getEffectiveModifiability
// Corresponding MHEG datatype: Effective-Modifiability
//==========================================================================
enum EffectiveModifiability {
    EFFECTIVELY_MODIFIABLE,
    EFFECTIVELY_NOT_MODIFIABLE
};
```

```
// Interface: RtCompositeOrStructuralSocket Operation: setResizingStrategy
// Interface: RtCompositeOrStructuralSocket Operation: getResizingStrategy
// Corresponding MHEG datatype: Resizing-Strategy
//=========================================================================
enum ResizingStrategy {
    FIXED,
    MINIMUM,
    GROWS_ONLY
};

// Interface: RtCompositeOrStructuralSocket Operation: setMenuInteractionStyle
// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Interface: RtContentOrPresentableSocket Operation: setSliderInteractionStyle
// Corresponding MHEG datatype: Orientation
//=========================================================================
enum Orientation {
    HORIZONTAL,
    VERTICAL
};

// Corresponding MHEG datatype: Presentation-Persistence
//=========================================================================
enum PresentationPersistence {
    PERSISTENT,
    NOT_PERSISTENT
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Corresponding MHEG datatype: Slider-Side
//=========================================================================
enum SliderSide {
    SIDE1,
    SIDE2
};

// Interface: RtGenericContentOrPresentableSocket Operation: setAudibleVolume
// Corresponding MHEG datatype: Audible-Volume
//=========================================================================
enum AudibleVolumeTag { SPECIFIED_VOLUME_TAG, LOGICAL_VOLUME_TAG,
IOV_SCALING_FACTOR_TAG, OV_SCALING_FACTOR_TAG };
union AudibleVolume
switch (AudibleVolumeTag){
    case SPECIFIED_VOLUME_TAG:
        long
            specified_volume;
    case LOGICAL_VOLUME_TAG:
        long
            logical_volume;
    case IOV_SCALING_FACTOR_TAG:
        long
            iov_scaling_factor;
    case OV_SCALING_FACTOR_TAG:
        long
            ov_scaling_factor;
};

// Interface: RtGenericContentOrPresentableSocket Operation:
//   setButtonInteractionStyle
// Corresponding MHEG datatype: Presentation-State
//=========================================================================
enum PresentationState {
    SELECTABLE_NOT_SELECTED,
    SELECTABLE_SELECTED,
    NOT_SELECTABLE_SELECTED,
    NOT_SELECTABLE_NOT_SELECTED
};

// Corresponding MHEG datatype: Echo-Mode
//=========================================================================
enum EchoMode {
    ITSELF,
    HIDDEN
};
```

```
// Interface: RtContentOrPresentableSocket Operation:
//   setEntryFieldInteractionStyle
// Corresponding MHEG datatype: Echo-Style
//==================================================================
enum EchoStyleTag { MODE_TAG, SPECIFIED_TAG };
union EchoStyle
switch (EchoStyleTag){
    case MODE_TAG:
        EchoMode
            mode;
    case SPECIFIED_TAG:
        string
            specified;
};

// Corresponding MHEG datatype: Channel-Reference
//==================================================================
enum ChannelReferenceTag { CHANNEL_IDENTIFIER_TAG, ALIAS_TAG,
NULL_CHANNEL_REFERENCE_TAG };
union ChannelReference
switch (ChannelReferenceTag){
    case CHANNEL_IDENTIFIER_TAG:
        ChannelIdentifier
            channel_identifier;
    case ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Interval
//==================================================================
struct Interval {
    sequence<long,1>
        start_point;
    sequence<long,1>
        end_point;
};

// Corresponding MHEG datatype: Generic-Volume-Range
//==================================================================
struct GenericVolumeRange {
    sequence<long,1>
        maximum_volume;
    sequence<long,1>
        minimum_volume;
};

// Interface: Channel Operation: new
// Corresponding MHEG datatype: Original-Def-Declaration
//==================================================================
struct OriginalDefDeclaration {
    sequence<long,1>
        generic_temporal_ratio;
    sequence<Interval,1>
        x_axis_interval;
    sequence<Interval,1>
        y_axis_interval;
    sequence<Interval,1>
        z_axis_interval;
    sequence<GenericVolumeRange,1>
        audible_volume_range_declaration;
};

// Interface: Channel Operation: getAvailability
// Corresponding MHEG datatype: Channel-Status-ValueCHANNEL-STATUS-VALUE-
//==================================================================
enum ChannelStatusValue {
    CHANNEL_STATUS_VALUE_AVAILABLE,
    CHANNEL_STATUS_VALUE_NOT_AVAILABLE,
    CHANNEL_STATUS_VALUE_PROCESSING
};
```

```
// Interface: Channel Operation: setPerceptability
// Interface: Channel Operation: getPerceptability
// Corresponding MHEG datatype: Channel-Perceptability-Values
//=========================================================================
enum ChannelPerceptabilityValue {
    ON,
    OFF
};

// Interface: NotificationManager Operation: getNotification
// Interface: MhContent Operation: getData
// Corresponding MHEG datatype: Generic-Value
//=========================================================================
enum GenericValueTag { BOOLEAN_FIELD_TAG, NUMERIC_TAG, STRING_FIELD_TAG,
GENERIC_LIST_TAG, UNSPECIFIED_TAG };
union GenericValue
switch (GenericValueTag){
    case BOOLEAN_FIELD_TAG:
        boolean
            boolean_field;
    case NUMERIC_TAG:
        long
            numeric;
    case STRING_FIELD_TAG:
        string
            string_field;
    case GENERIC_LIST_TAG:
        sequence<GenericValue>
            generic_list;
};

// Corresponding MHEG datatype: Generic-String
//=========================================================================
enum GenericStringTag { GENERIC_STRING_CONSTANT_TAG,
GENERIC_STRING_UNSPECIFIED_TAG };
union GenericString
switch (GenericStringTag){
    case GENERIC_STRING_CONSTANT_TAG:
        string
           constant;
};

// Interface: Socket Operation: setVisibleDurationPosition
// Corresponding MHEG datatype: Visible-Duration
//=========================================================================
enum VisibleDurationPositionTag {
VISIBLE_DURATION_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG,
VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union VisibleDurationPosition
switch (VisibleDurationPositionTag){
    case VISIBLE_DURATION_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
           specified_temporal_point;
    case VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
           logical_temporal_PD_point;
    case VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
           logical_temporal_VD_point;
};

// Interface: RtComponentOrSocket Operation: getRGS
// Corresponding MHEG datatype: none
//=========================================================================
enum RGSValueTag { RGS_VALUE_CHANNEL_IDENTIFIER_TAG, RGS_VALUE_NULL_CHANNEL_TAG,
RGS_VALUE_PRGS_TAG };
union RGSValue
switch (RGSValueTag){
    case RGS_VALUE_CHANNEL_IDENTIFIER_TAG:
        ChannelIdentifier
           channel_identifier;
};
```

```
// Corresponding MHEG datatype: Generic-Numeric
//======================================================================
enum GenericNumericTag { GENERIC_NUMERIC_CONSTANT_TAG,
GENERIC_NUMERIC_UNSPECIFIED_TAG };
union GenericNumeric
switch (GenericNumericTag){
    case GENERIC_NUMERIC_CONSTANT_TAG:
        long
            constant;
};

// Interface: RtComponentOrSocket Operation: getSelectionMode
// Corresponding MHEG datatype: none
//======================================================================
enum SelectionModeValueTag { USER_INTERACTION_TAG, NO_SELECTION_TAG,
MHEG_ACTION_TAG, USING_APPLICATION_ACTION_TAG };
union SelectionModeValue
switch (SelectionModeValueTag){
    case USER_INTERACTION_TAG:
        unsigned long
            user_interaction;
};

// Interface: RtComponentOrSocket Operation: getModificationMode
// Corresponding MHEG datatype: none
//======================================================================
enum ModificationModeValueTag { MODIFICATION_MODE_VALUE_USER_INTERACTION_TAG,
MODIFICATION_MODE_VALUE_NO_MODIFICATION_TAG,
MODIFICATION_MODE_VALUE_MHEG_ACTION_TAG,
MODIFICATION_MODE_VALUE_USING_APPLICATION_ACTION_TAG,
MODIFICATION_MODE_VALUE_CHILD_TAG };
union ModificationModeValue
switch (ModificationModeValueTag){
    case MODIFICATION_MODE_VALUE_USER_INTERACTION_TAG:
        unsigned long
            user_interaction;
};

// Corresponding MHEG datatype: External-Identifier
//======================================================================
enum ExternalIdentifierTag { EXTERNAL_LONG_ID_TAG, PUBLIC_ID_TAG,
SYSTEM_ID_TAG };
union ExternalIdentifier
switch (ExternalIdentifierTag){
    case EXTERNAL_LONG_ID_TAG:
        ExternalLongIdentifier
            external_long_id;
    case PUBLIC_ID_TAG:
        PublicIdentifier
            public_id;
    case SYSTEM_ID_TAG:
        SystemIdentifier
            system_id;
};

// Corresponding MHEG datatype: Container-Tail
//======================================================================
struct ContainerTail {
    sequence<long>
        indexes;
    enum ContainerTailTag { INDEX_TAG, CONTAINER_CHILD_REF_TAG } tag;
    union ContainerTail
    switch (ContainerTailTag){
        case INDEX_TAG:
            long
                index;
        case CONTAINER_CHILD_REF_TAG:
            ContainerChildReference
                container_child_ref;
    } end;
};
```

```
// Corresponding MHEG datatype: Specified-Sizes
//======================================================================
struct SpecifiedSizes {
    sequence<GenericNumeric,1>
        x_axis_length;
    sequence<GenericNumeric,1>
        y_axis_length;
    sequence<GenericNumeric,1>
        z_axis_length;
};

// Corresponding MHEG datatype: Attachment-Attribute
//======================================================================
enum AttachmentAttributeTag { SPECIFIED_POSITION_TAG, LOGICAL_POSITION_TAG };
union AttachmentAttribute
switch (AttachmentAttributeTag){
    case SPECIFIED_POSITION_TAG:
        GenericNumeric
            specified_position;
    case LOGICAL_POSITION_TAG:
        GenericNumeric
            logical_position;
};

// Corresponding MHEG datatype: Length-Attribute
//======================================================================
enum LengthAttributeTag { SPECIFIED_LENGTH_TAG, RELATIVE_LENGTH_TAG };
union LengthAttribute
switch (LengthAttributeTag){
    case SPECIFIED_LENGTH_TAG:
        GenericNumeric
            specified_length;
    case RELATIVE_LENGTH_TAG:
        GenericNumeric
            relative_length;
};

// Interface: RtComponentOrSocket Operation: getPS
// Interface: RtComponentOrSocket Operation: getPSAP
// Interface: RtComponentOrSocket Operation: getVS
// Interface: RtComponentOrSocket Operation: getVSIAP
// Interface: RtComponentOrSocket Operation: getVSIAPPosition
// Interface: RtComponentOrSocket Operation: getVSEAP
// Interface: RtComponentOrSocket Operation: getVSEAPPosition
// Corresponding MHEG datatype: Specified-Position
//======================================================================
struct SpecifiedPosition {
    GenericNumeric
        x_point;
    GenericNumeric
        y_point;
    GenericNumeric
        z_point;
};

// Interface: RtComponentOrSocket Operation: setPresentationPriority
// Corresponding MHEG datatype: Presentation-Priority
//======================================================================
enum PresentationPriorityTag { GENERIC_NUMERIC_TAG, PRIORITY_LEVEL_TAG };
union PresentationPriority
switch (PresentationPriorityTag){
    case GENERIC_NUMERIC_TAG:
        GenericNumeric
            generic_numeric;
    case PRIORITY_LEVEL_TAG:
        PriorityLevel
            priority_level;
};
```

```
// Interface: RtComponentOrSocket Operation: setTimestones
// Corresponding MHEG datatype: Timestone
//=======================================================================
struct Timestone {
    long
        timestone_identifier;
    TimestonePosition
        timestone_position;
};

// Interface: RtComponentOrSocket Operation: setVisibleSize
// Corresponding MHEG datatype: none
//=======================================================================
enum VSTag { X_SIZE_ATTRIBUTE_TAG, Y_SIZE_ATTRIBUTE_TAG, Z_SIZE_ATTRIBUTE_TAG };
union VS
switch (VSTag){
    case X_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            x_size_attribute;
    case Y_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            y_size_attribute;
    case Z_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            z_size_attribute;
};

// Interface: RtComponentOrSocket Operation: setAttachmentPoint
// Corresponding MHEG datatype: none
//=======================================================================
struct AttachmentPoint {
    sequence<AttachmentAttribute,1>
        x_attachment;
    sequence<AttachmentAttribute,1>
        y_attachment;
    sequence<AttachmentAttribute,1>
        z_attachment;
};

// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Lengths
//=======================================================================
struct Lengths {
    sequence<LengthAttribute,1>
        x_length;
    sequence<LengthAttribute,1>
        y_length;
    sequence<LengthAttribute,1>
        z_length;
};

// Interface: RtMultiplexedContentOrPresentableSocket Operation: getStreamChosen
// Corresponding MHEG datatype: none
//=======================================================================
enum StreamValueTag { STREAM_IDENTIFIER_TAG, NO_STREAM_CHOSEN_TAG };
union StreamValue
switch (StreamValueTag){
    case STREAM_IDENTIFIER_TAG:
        StreamIdentifier
            stream_identifier;
};

// Interface: MhContent Operation: setData
// Corresponding MHEG datatype: Data-Element
//=======================================================================
struct DataElement {
    sequence<long>
        element_list_index;
    GenericValue
        generic_value;
};
```

```
// Interface: NotificationManager Operation: getNotification
// Interface: MhObject Operation: bind
// Interface: MhObject Operation: prepare
// Interface: MhGenericContent Operation: copy
// Corresponding MHEG datatype: Mh-Object-Reference
//==========================================================================
struct MhObjectReference {
enum MhObjectReferenceHeadTag { MHEG_IDENTIFIER_TAG, EXTERNAL_IDENTIFIER_TAG,
ALIAS_TAG, NULL_OBJECT_REF_TAG } head_tag;
union MhObjectReferenceHead
switch (MhObjectReferenceHeadTag){
    case MHEG_IDENTIFIER_TAG:
        MHEGIdentifier
            mheg_identifier;
    case EXTERNAL_IDENTIFIER_TAG:
        ExternalIdentifier
            external_identifier;
    case ALIAS_TAG:
        Alias
            alias;
} head;
enum MhObjectReferenceTailTag { CONTAINER_ELEMENT_REFERENCE_TAG,
OTHER_REFERENCE_TAG } tail_tag;
union MhObjectReferenceTail
switch (MhObjectReferenceTailTag){
    case CONTAINER_ELEMENT_REFERENCE_TAG:
        ContainerTail
            container_tail;
} tail;
};

// Interface: RtComponentOrSocket Operation: setPerceptibleSizeProjection
// Corresponding MHEG datatype: Perceptible-Size-Projection
//==========================================================================
struct PerceptibleSizeProjection {
    sequence<PerceptibleProjection,1>
        x_perceptible_size_projection;
    sequence<PerceptibleProjection,1>
        y_perceptible_size_projection;
    sequence<PerceptibleProjection,1>
        z_perceptible_size_projection;
};

// Corresponding MHEG datatype: Rt-Object-Number-Reference
//==========================================================================
enum RtObjectNumberReferenceTag { RT_OBJECT_NUMBER_TAG, RT_DYNAMIC_REFERENCE_TAG
};
union RtObjectNumberReference
switch (RtObjectNumberReferenceTag){
    case RT_OBJECT_NUMBER_TAG:
        long
            rt_object_number;
    case RT_DYNAMIC_REFERENCE_TAG:
        RtDynamicReference
            rt_dynamic_reference;
};

// Interface: RtObject Operation: bind
// Interface: RtObject Operation: new
// Corresponding MHEG datatype: Rt-Object-Reference
//==========================================================================
struct RtObjectReference {
    MhObjectReference
        model_object_reference;
    RtObjectNumberReference
        rt_object_number_reference;
};
```

```
// Corresponding MHEG datatype: Rt-Reference
//============================================================================
enum RtReferenceTag { RT_REFERENCE_RT_OBJECT_REFERENCE_TAG,
RT_REFERENCE_ALIAS_TAG, RT_REFERENCE_NULL_RT_OBJECT_TAG };
union RtReference
switch (RtReferenceTag){
    case RT_REFERENCE_RT_OBJECT_REFERENCE_TAG:
        RtObjectReference
            rt_object_reference;
    case RT_REFERENCE_ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Socket-Tail
//============================================================================
struct SocketTail {
    sequence<long>
        indexes;
    enum SocketTailTag { INDEX_TAG, SUB_SOCKET_REF_TAG } tag;
    union SocketTail
    switch (SocketTailTag){
        case INDEX_TAG:
            long
                index;
        case SUB_SOCKET_REF_TAG:
            SubSocketReference
                sub_socket_ref;
    } end;
};

// Corresponding MHEG datatype: Indexed-Child-Socket
//============================================================================
struct IndexedChildSocket {
    long
        index;
    SocketTail
        tail;
};

// Interface: Socket Operation: bind
// Interface: Socket Operation: getIdentification
// Corresponding MHEG datatype: Socket-Identification
//============================================================================
struct SocketIdentification {
    RtReference
        rt_composite_reference;
    SocketTail
        socket_tail;
};

// Interface: Socket Operation: bind
// Corresponding MHEG datatype: Socket-Reference
//============================================================================
enum SocketReferenceTag { SOCKET_REFERENCE_SOCKET_IDENT_TAG,
SOCKET_REFERENCE_ALIAS_TAG };
union SocketReference
switch (SocketReferenceTag){
    case SOCKET_REFERENCE_SOCKET_IDENT_TAG:
        SocketIdentification
            socket_ident;
    case SOCKET_REFERENCE_ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Rt-Object-Socket-Reference
//============================================================================
enum RtObjectSocketReferenceTag { RT_REFERENCE_TAG, SOCKET_REFERENCE_TAG };
union RtObjectSocketReference
switch (RtObjectSocketReferenceTag){
    case RT_REFERENCE_TAG:
        RtReference
            rt_reference;
    case SOCKET_REFERENCE_TAG:
        SocketReference
            socket_reference;
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Interface: RtContentOrPresentableSocket Operation: setSliderInteractionStyle
// Interface: RtContentOrPresentableSocket Operation:
//   setEntryFieldInteractionStyle
// Interface: Channel Operation: getAssignedPerceptibles
// Corresponding MHEG datatype: Perceptible-Reference
//===========================================================================
enum PerceptibleReferenceTag { RT_COMPONENT_REFERENCE_TAG,
RT_SOCKET_REFERENCE_TAG };
union PerceptibleReference
switch (PerceptibleReferenceTag){
    case RT_COMPONENT_REFERENCE_TAG:
        RtReference
            rt_component_reference;
    case RT_SOCKET_REFERENCE_TAG:
        SocketReference
            rt_socket_reference;
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Corresponding MHEG datatype: Separator
//===========================================================================
enum SeparatorTag { NO_TAG, YES_DEFAULT_TAG, SEPARATOR_PIECE_TAG };
union Separator
switch (SeparatorTag){
    case SEPARATOR_PIECE_TAG:
        PerceptibleReference
            separator_piece;
};

// Interface: RtCompositeOrStructuralSocket Operation: setMenuInteractionStyle
// Corresponding MHEG datatype: Association
//===========================================================================
struct Association {
    sequence<SocketReference,1>
        title;
    sequence<Separator,1>
        separator;
    sequence<SocketReference,1>
        submenu;
    sequence<PresentationPersistence,1>
        submenu_presentation_persistence;
    sequence<Orientation,1>
        submenu_orientation;
};

// Interface: RtSocket Operation: plug
// Corresponding MHEG datatype: Plug-In
//===========================================================================
enum PlugInTag { PLUG_IN_RT_COMPONENT_REFERENCE_TAG,
PLUG_IN_COMPONENT_REFERENCE_TAG, PLUG_IN_LABEL_TAG };
union PlugIn
switch (PlugInTag){
    case PLUG_IN_RT_COMPONENT_REFERENCE_TAG:
        RtObjectReference
            rt_component_reference;
    case PLUG_IN_COMPONENT_REFERENCE_TAG:
        MhObjectReference
            component_reference;
    case PLUG_IN_LABEL_TAG:
        GenericString
            label;
};
```

```
// Interface: RtComponentOrSocket Operation: getVSEAPPosition
// Corresponding MHEG datatype: none
//==========================================================================
enum ReferencePointTag { VSEAP_POSITION_ORIGIN_RGS_TAG,
VSEAP_POSITION_ORIGIN_CGS_TAG, VSEAP_POSITION_SAME_RGS_COMPONENT_TAG,
VSEAP_POSITION_SAME_CGS_COMPONENT_TAG, VSEAP_POSITION_SPECIFIED_POSITION_TAG };
union ReferencePoint
switch (ReferencePointTag){
    case VSEAP_POSITION_SAME_RGS_COMPONENT_TAG:
        RtObjectSocketReference
            same_RGS_component;
    case VSEAP_POSITION_SAME_CGS_COMPONENT_TAG:
        RtObjectSocketReference
            same_CGS_component;
    case VSEAP_POSITION_SPECIFIED_POSITION_TAG:
        SpecifiedPosition
            specified_position;
};

// Interface: RtScript Operation: setParameters
// Corresponding MHEG datatype: Parameter
//==========================================================================
enum ParameterTag { GENERIC_VALUE_TAG, MH_OBJECT_REFERENCE_TAG };
union Parameter
switch (ParameterTag){
    case GENERIC_VALUE_TAG:
        GenericValue
            generic_value;
    case MH_OBJECT_REFERENCE_TAG:
        MhObjectReference
            mh_object_reference;
};

// Interface: RtObjectOrSocket Operation: setGlobalBehaviour
// Corresponding MHEG datatype: Global-Behaviour
//==========================================================================
enum GlobalBehaviourTag { GLOBAL_BEHAVIOUR_RT_REFERENCE_TAG,
GLOBAL_BEHAVIOUR_GENERIC_LIST_TAG, GLOBAL_BEHAVIOUR_UNSPECIFIED_TAG };
union GlobalBehaviour
switch (GlobalBehaviourTag){
    case GLOBAL_BEHAVIOUR_RT_REFERENCE_TAG:
        RtReference
            rt_reference;
    case GLOBAL_BEHAVIOUR_GENERIC_LIST_TAG:
        GenericValue
            generic_list;
};

// Interface: RtComponentOrSocket Operation: setVisibleSizesAdjustment
// Corresponding MHEG datatype: Adjustment-PolicyADJUSTMENT-POLICY-
//==========================================================================
enum AdjustmentPolicyTag { ADJUSTMENT_POLICY_COMPONENT_REFERENCE_TAG,
ADJUSTMENT_POLICY_SPECIFIED_TAG, ADJUSTMENT_POLICY_GREATEST_TAG,
ADJUSTMENT_POLICY_SMALLEST_TAG };
union AdjustmentPolicy
switch (AdjustmentPolicyTag){
    case ADJUSTMENT_POLICY_COMPONENT_REFERENCE_TAG:
        RtObjectSocketReference
            component_reference;
    case ADJUSTMENT_POLICY_SPECIFIED_TAG:
        SpecifiedSizes
            specified;
};
```

```
// Interface: RtObject Operation: bind
// Interface: RtObject Operation: new
// Interface: RtObject Operation: getIdentifier
// Corresponding MHEG datatype: none
//==========================================================================
struct RtObjectIdentifier {
    MHEGIdentifier
        model_object_id;
    long
        rt_object_number;
};
```

```
// Interface: RtGenericContentOrPresentableSocket Operation:
//   setButtonInteractionStyle
// Corresponding MHEG datatype: Alternate-Presentation-State
//==========================================================================
struct AlternatePresentation {
    PresentationState
        presentation_state;
    PerceptibleReference
        perceptible_target;
};
```

7.1.37 Exceptions

7.1.37.1 `InvalidTarget` exception

Description:

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current period of the target.

7.1.37.2 `InvalidParameter` exception

Description:

The `InvalidParameter` exception is raised when the value of one of the parameters prohibits the normal execution of the action. The `completion_status` member indicates whether the action was completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

7.1.37.3 `NotBound` exception

Description:

The `NotBound` exception is raised when the interface object instance is not bound with an MHEG entity.

7.1.37.4 `AlreadyBound` exception

Description:

The `AlreadyBound` exception is raised when the interface object instance is already bound with an MHEG entity. The `entity_identifier` member identifies the bound entity.

7.1.37.5 IDL definition

```
exception InvalidTarget {
    unsigned short period;
};

enum CompletionStatus { YES, NO};

exception InvalidParameter {
    CompletionStatus completion_status;
    unsigned short period;
};

typedef long EntityIdentifier
exception AlreadyBound {
    EntityIdentifier entity_identifier;
};

exception NotBound {};
```

7.2 Optional primitives

The following objects shall be used to modify MHEG objects in form b:

- `mhAction`
- `mhComposite`
- `mhContainer`
- `mhContent`
- `mhDescriptor`
- `mhLink`
- `mhMultiplexedContent`
- `mhScript`

Annex A

Complete IDL definition of the MHEG API

```
//==========================================================================
typedef sequence<long> ApplicationIdentifier;

// Corresponding MHEG datatype: Object-Number
//==========================================================================
typedef long ObjectNumber;

// Interface: MhObject Operation: bind
// Interface: MhObject Operation: prepare
// Interface: MhObject Operation: getIdentifier
// Corresponding MHEG datatype: MHEG-Identifier
//==========================================================================
struct MHEGIdentifier {
    sequence<ApplicationIdentifier,1>
        application_identifier;
    ObjectNumber
        object_number;
};

// Corresponding MHEG datatype: Public-Identifier
//==========================================================================
typedef string PublicIdentifier;

// Corresponding MHEG datatype: System-Identifier
//==========================================================================
typedef string SystemIdentifier;

// Corresponding MHEG datatype: External-Long-Identifier
//==========================================================================
struct ExternalLongIdentifier {
    PublicIdentifier
        public_identifier;
    SystemIdentifier
        system_identifier;
};

// Corresponding MHEG datatype: Alias
//==========================================================================
typedef string Alias;

// Corresponding MHEG datatype: Container-Child-Reference
//==========================================================================
enum ContainerChildReference {
    CHILD,
    DESCENDANT
};
```

```
// Interface: MhObject Operation: getPreparationStatus
// Corresponding MHEG datatype: Preparation-Status-Value
//===========================================================================
enum PreparationStatusValue {
    READY,
    NOT_READY,
    PROCESSING
};

// Interface: MhMultiplexedContent Operation: setMultiplex
// Interface: MhMultiplexedContent Operation: setDemultiplex
// Interface: RtMultiplexedContentOrPresentableSocket Operation: setStreamChoice
// Corresponding MHEG datatype: Stream-Identifier
//===========================================================================
typedef sequence<long> StreamIdentifier;

// Corresponding MHEG datatype: Rt-Dynamic-Reference
//===========================================================================
enum RtDynamicReference {
    QUESTION_MARK,
    STAR
};

// Interface: RtObject Operation: getAvailabilityStatus
// Corresponding MHEG datatype: Rt-Availibility-Status-Value
//===========================================================================
enum RtAvailabilityStatusValue {
    RT_AVAILABILITY_STATUS_VALUE_AVAILABLE,
    RT_AVAILABILITY_STATUS_VALUE_NOT_AVAILABLE,
    RT_AVAILABILITY_STATUS_VALUE_PROCESSING
};

// Interface: RtObject Operation: getRunningStatus
// Corresponding MHEG datatype: Running-Status-Value
//===========================================================================
enum RunningStatusValue {
    RUNNING_STATUS_VALUE_RUNNING,
    RUNNING_STATUS_VALUE_NOT_RUNNING,
    RUNNING_STATUS_VALUE_PROCESSING
};

// Interface: RtScript Operation: getTerminationStatus
// Corresponding MHEG datatype: Termination-Status-Value
//===========================================================================
enum TerminationStatusValue {
    TERMINATED,
    NOT_TERMINATED
};

// Interface: RtComponentOrSocket Operation: setRGS
// Interface: Channel Operation: getIdentifier
// Corresponding MHEG datatype: Channel-Identifier
//===========================================================================
typedef long ChannelIdentifier;

// Corresponding MHEG datatype: Priority-Level
//===========================================================================
enum PriorityLevel {
    INCREMENT_PRIORITY,
    DECREMENT_PRIORITY
};

// Interface: RtComponentOrSocket Operation: setVisibleDuration
// Corresponding MHEG datatype: Temporal-Position
//===========================================================================
enum TemporalPositionTag { SPECIFIED_TEMPORAL_POINT_TAG,
LOGICAL_TEMPORAL_PD_POINT_TAG };
union TemporalPosition
switch (TemporalPositionTag){
    case SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
            logical_temporal_PD_point;
};
// Interface: RtComponentOrSocket Operation: setCurrentTemporalPosition
// Corresponding MHEG datatype: Current-Temporal-Position
//==========================================================================
enum CurrentTemporalPositionTag {
CURRENT_TEMPORAL_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
CURRENT_TEMPORAL_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union CurrentTemporalPosition
switch (CurrentTemporalPositionTag){
    case CURRENT_TEMPORAL_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case CURRENT_TEMPORAL_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
            logical_temporal_vd_point;
};
// Interface: RtComponentOrSocket Operation: setTemporalTermination
// Interface: RtComponentOrSocket Operation: getTemporalTermination
// Corresponding MHEG datatype: Temporal-Termination
//==========================================================================
enum TemporalTermination {
    TEMPORAL_TERMINATION_FREEZE,
    TEMPORAL_TERMINATION_STOP
};
// Interface: RtComponentOrSocket Operation: setSpeed
// Corresponding MHEG datatype: Speed
//==========================================================================
enum SpeedTag { SPECIFIED_OGTR_TAG, SPEED_RATE_TAG, SCALING_FACTOR_TAG };
union Speed
switch (SpeedTag){
    case SPECIFIED_OGTR_TAG:
        long
            specified_OGTR;
    case SPEED_RATE_TAG:
        long
            speed_rate;
    case SCALING_FACTOR_TAG:
        long
            scaling_factor;
};
// Corresponding MHEG datatype: Timestone-Position
//==========================================================================
enum TimestonePositionTag { TIMESTONE_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
TIMESTONE_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG,
TIMESTONE_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union TimestonePosition
switch (TimestonePositionTag){
    case TIMESTONE_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case TIMESTONE_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
            logical_temporal_PD_point;
    case TIMESTONE_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
            logical_temporal_VD_point;
};


// Interface: RtComponentOrSocket Operation: getVDLength
// Corresponding MHEG datatype: GT-Indicator
//==========================================================================
enum GTIndicator {
      OGTU,
      RGTU
};
```

```
// Corresponding MHEG datatype: Perceptible-Projection
//==========================================================================
enum PerceptibleProjectionTag { SPECIFIED_SIZE_TAG, IOGSR_SCALING_FACTOR_TAG,
COGSR_SCALING_FACTOR_TAG };
union PerceptibleProjection
switch (PerceptibleProjectionTag){
    case SPECIFIED_SIZE_TAG:
        long
            specified_size;
    case IOGSR_SCALING_FACTOR_TAG:
        long
            iogsr_scaling_factor;
    case COGSR_SCALING_FACTOR_TAG:
        long
            cogsr_scaling_factor;
};

// Interface: RtComponentOrSocket Operation: setAspectRatioPreserved
// Interface: RtComponentOrSocket Operation: getAspectRatio
// Corresponding MHEG datatype: Aspect-Ratio
//==========================================================================
enum AspectRatio {
    PRESERVED,
    NOT_PRESERVED
};

// Interface: RtComponentOrSocket Operation: setVisibleSize
// Interface: RtComponentOrSocket Operation: getVSGS
// Corresponding MHEG datatype: VSGS
//==========================================================================
enum VSGS {
    THIS,
    RELATIVE
};

// Corresponding MHEG datatype: Size-Attribute
//==========================================================================
enum SizeAttributeTag { SIZE_ATTRIBUTE_SPECIFIED_SIZE_TAG,
SIZE_ATTRIBUTE_IVS_RELATIVE_TAG,
SIZE_ATTRIBUTE_CVS_RELATIVE_TAG };
union SizeAttribute
switch (SizeAttributeTag){
    case SIZE_ATTRIBUTE_SPECIFIED_SIZE_TAG:
        long
            specified_size;
    case SIZE_ATTRIBUTE_IVS_RELATIVE_TAG:
        long
            ivs_relative;
    case SIZE_ATTRIBUTE_CVS_RELATIVE_TAG:
        long
            cvs_relative;
};

// Interface: RtComponentOrSocket Operation: setVisibleSizesAdjustment
// Corresponding MHEG datatype: Adjustment-Axis
//==========================================================================
enum AdjustmentAxis {
    X_AXIS,
    Y_AXIS,
    Z_AXIS
};

// Corresponding MHEG datatype: Sub-Socket-Reference
//==========================================================================
enum SubSocketReference {
    SUB_SOCKET_REFERENCE_CHILD,
    SUB_SOCKET_REFERENCE_DESCENDANT,
    SUB_SOCKET_REFERENCE_QUESTION_MARK_CHILD,
    SUB_SOCKET_REFERENCE_QUESTION_MARK_DESCENDANT
};
```

```
// Interface: RtComponentOrSocket Operation: setBox
// Interface: RtComponentOrSocket Operation: getBox
// Corresponding MHEG datatype: Box-Constants
//==========================================================================
enum BoxConstants {
    PRESENTED,
    NOT_PRESENTED
};

// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Reference-Type
//==========================================================================
enum ReferenceType {
    VSIAP,
    VSEAP
};

// Interface: RtComponentOrSocket Operation: setAttachmentPoint
// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Attachment-Point-Type
//==========================================================================
enum AttachmentPointType {
    ATTACHMENT_POINT_TYPE_PSAP,
    ATTACHMENT_POINT_TYPE_VSIAP,
    ATTACHMENT_POINT_TYPE_VSEAP
};

// Interface: RtComponentOrSocket Operation: setVisibleSizesAlignment
// Corresponding MHEG datatype: Size-Border
//==========================================================================
enum SizeBorder {
    TOP,
    BOTTOM,
    RIGHT,
    LEFT,
    UPPER_Z,
    LOWER_Z,
    CENTER_X,
    CENTER_Y,
    CENTER_Z
};

// Interface: RtComponentOrSocket Operation: setMovingAbility
// Interface: RtComponentOrSocket Operation: setResizingAbility
// Interface: RtComponentOrSocket Operation: setScalingAbility
// Interface: RtComponentOrSocket Operation: setScrollingAbility
// Interface: RtComponentOrSocket Operation: getMovingAbility
// Interface: RtComponentOrSocket Operation: getResizingAbility
// Interface: RtComponentOrSocket Operation: getScalingAbility
// Interface: RtComponentOrSocket Operation: getScrollingAbility
// Corresponding MHEG datatype: User-Controls
//==========================================================================
enum UserControls {
    ALLOWED,
    NOT_ALLOWED
};

// Interface: RtComponentOrSocket Operation: getPS
// Corresponding MHEG datatype: GS-Indicator
//==========================================================================
enum GSIndicator {
    OGSU,
    RGSU
};

// Interface: RtComponentOrSocket Operation: getPSAP
// Interface: RtComponentOrSocket Operation: getVSIAP
// Corresponding MHEG datatype: Point-Type
//==========================================================================
enum PointType {
    RELATIVE_POINT,
    ABSOLUTE_POINT
};
```

```
// Interface: RtComponentOrSocket Operation: setSelectionStatus
// Interface: RtComponentOrSocket Operation: getSelectionStatus
// Corresponding MHEG datatype: Selection-Status-Value
//==========================================================================
enum SelectionStatusValue {
    SELECTED,
    NOT_SELECTED
};

// Interface: RtComponentOrSocket Operation:
//   setSelectionPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   getSelectionPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   setModificationPresentationEffectResponsibility
// Interface: RtComponentOrSocket Operation:
//   getModificationPresentationEffectResponsibility
// Corresponding MHEG datatype: Responsibility
//==========================================================================
enum Responsibility {
    MHEG_ENGINE,
    AUTHOR
};

// Interface: RtComponentOrSocket Operation: getEffectiveSelectability
// Corresponding MHEG datatype: Effective-Selectability
//==========================================================================
enum EffectiveSelectability {
    EFFECTIVELY_SELECTABLE,
    EFFECTIVELY_NOT_SELECTABLE
};

// Interface: RtComponentOrSocket Operation: setModificationStatus
// Interface: RtComponentOrSocket Operation: getModificationStatus
// Corresponding MHEG datatype: Modification-Status-Value
//==========================================================================
enum ModificationStatusValue {
    MODIFIED,
    MODIFYING,
    NOT_MODIFIED
};

// Interface: RtComponentOrSocket Operation: getEffectiveModifiability
// Corresponding MHEG datatype: Effective-Modifiability
//==========================================================================
enum EffectiveModifiability {
    EFFECTIVELY_MODIFIABLE,
    EFFECTIVELY_NOT_MODIFIABLE
};

// Interface: RtCompositeOrStructuralSocket Operation: setResizingStrategy
// Interface: RtCompositeOrStructuralSocket Operation: getResizingStrategy
// Corresponding MHEG datatype: Resizing-Strategy
//==========================================================================
enum ResizingStrategy {
    FIXED,
    MINIMUM,
    GROWS_ONLY
};

// Interface: RtCompositeOrStructuralSocket Operation: setMenuInteractionStyle
// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Interface: RtContentOrPresentableSocket Operation: setSliderInteractionStyle
// Corresponding MHEG datatype: Orientation
//==========================================================================
enum Orientation {
    HORIZONTAL,
    VERTICAL
};
```

```
// Corresponding MHEG datatype: Presentation-Persistence
//========================================================================
enum PresentationPersistence {
    PERSISTENT,
    NOT_PERSISTENT
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Corresponding MHEG datatype: Slider-Side
//========================================================================
enum SliderSide {
    SIDE1,
    SIDE2
};

// Interface: RtGenericContentOrPresentableSocket Operation: setAudibleVolume
// Corresponding MHEG datatype: Audible-Volume
//========================================================================
enum AudibleVolumeTag { SPECIFIED_VOLUME_TAG, LOGICAL_VOLUME_TAG,
IOV_SCALING_FACTOR_TAG,
OV_SCALING_FACTOR_TAG };
union AudibleVolume
switch (AudibleVolumeTag){
    case SPECIFIED_VOLUME_TAG:
        long
            specified_volume;
    case LOGICAL_VOLUME_TAG:
        long
            logical_volume;
    case IOV_SCALING_FACTOR_TAG:
        long
            iov_scaling_factor;
    case OV_SCALING_FACTOR_TAG:
        long
            ov_scaling_factor;
};

// Interface: RtGenericContentOrPresentableSocket Operation:
//   setButtonInteractionStyle
// Corresponding MHEG datatype: Presentation-State
//========================================================================
enum PresentationState {
    SELECTABLE_NOT_SELECTED,
    SELECTABLE_SELECTED,
    NOT_SELECTABLE_SELECTED,
    NOT_SELECTABLE_NOT_SELECTED
};

// Corresponding MHEG datatype: Echo-Mode
//========================================================================
enum EchoMode {
    ITSELF,
    HIDDEN
};

// Interface: RtContentOrPresentableSocket Operation:
//   setEntryFieldInteractionStyle
// Corresponding MHEG datatype: Echo-Style
//========================================================================
enum EchoStyleTag { MODE_TAG, SPECIFIED_TAG };
union EchoStyle
switch (EchoStyleTag){
    case MODE_TAG:
        EchoMode
            mode;
    case SPECIFIED_TAG:
        string
            specified;
};
```

```
// Corresponding MHEG datatype: Channel-Reference
//=================================================================
enum ChannelReferenceTag { CHANNEL_IDENTIFIER_TAG, ALIAS_TAG,
NULL_CHANNEL_REFERENCE_TAG };
union ChannelReference
switch (ChannelReferenceTag){
    case CHANNEL_IDENTIFIER_TAG:
        ChannelIdentifier
            channel_identifier;
    case ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Interval
//=========================================================================
struct Interval {
    sequence<long,1>
        start_point;
    sequence<long,1>
        end_point;
};

// Corresponding MHEG datatype: Generic-Volume-Range
//=========================================================================
struct GenericVolumeRange {
    sequence<long,1>
        maximum_volume;
    sequence<long,1>
        minimum_volume;
};

// Interface: Channel Operation: new
// Corresponding MHEG datatype: Original-Def-Declaration
//=========================================================================
struct OriginalDefDeclaration {
    sequence<long,1>
        generic_temporal_ratio;
    sequence<Interval,1>
        x_axis_interval;
    sequence<Interval,1>
        y_axis_interval;
    sequence<Interval,1>
        z_axis_interval;
    sequence<GenericVolumeRange,1>
        audible_volume_range_declaration;
};

// Interface: Channel Operation: getAvailability
// Corresponding MHEG datatype: Channel-Status-ValueCHANNEL-STATUS-VALUE-
//=========================================================================
enum ChannelStatusValue {
    CHANNEL_STATUS_VALUE_AVAILABLE,
    CHANNEL_STATUS_VALUE_NOT_AVAILABLE,
    CHANNEL_STATUS_VALUE_PROCESSING
};

// Interface: Channel Operation: setPerceptability
// Interface: Channel Operation: getPerceptability
// Corresponding MHEG datatype: Channel-Perceptability-Values
//=========================================================================
enum ChannelPerceptabilityValue {
    ON,
    OFF
};

// Interface: NotificationManager Operation: getNotification
// Interface: MhContent Operation: getData
// Corresponding MHEG datatype: Generic-Value
//=========================================================================
enum GenericValueTag { BOOLEAN_FIELD_TAG, NUMERIC_TAG, STRING_FIELD_TAG,
GENERIC_LIST_TAG,
UNSPECIFIED_TAG };
union GenericValue
```

```
switch (GenericValueTag){
    case BOOLEAN_FIELD_TAG:
        boolean
            boolean_field;
    case NUMERIC_TAG:
        long
            numeric;
    case STRING_FIELD_TAG:
        string
            string_field;
    case GENERIC_LIST_TAG:
        sequence<GenericValue>
            generic_list;
};

// Corresponding MHEG datatype: Generic-String
//==========================================================================
enum GenericStringTag { GENERIC_STRING_CONSTANT_TAG,
GENERIC_STRING_UNSPECIFIED_TAG };
union GenericString
switch (GenericStringTag){
    case GENERIC_STRING_CONSTANT_TAG:
        string
            constant;
};

// Interface: Socket Operation: setVisibleDurationPosition
// Corresponding MHEG datatype: Visible-Duration
//==========================================================================
enum VisibleDurationPositionTag {
VISIBLE_DURATION_POSITION_SPECIFIED_TEMPORAL_POINT_TAG,
VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG,
VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG };
union VisibleDurationPosition
switch (VisibleDurationPositionTag){
    case VISIBLE_DURATION_POSITION_SPECIFIED_TEMPORAL_POINT_TAG:
        long
            specified_temporal_point;
    case VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_PD_POINT_TAG:
        long
            logical_temporal_PD_point;
    case VISIBLE_DURATION_POSITION_LOGICAL_TEMPORAL_VD_POINT_TAG:
        long
            logical_temporal_VD_point;
};

// Interface: RtComponentOrSocket Operation: getRGS
// Corresponding MHEG datatype: none
//==========================================================================
enum RGSValueTag { RGS_VALUE_CHANNEL_IDENTIFIER_TAG, RGS_VALUE_NULL_CHANNEL_TAG,
RGS_VALUE_PRGS_TAG };
union RGSValue
switch (RGSValueTag){
    case RGS_VALUE_CHANNEL_IDENTIFIER_TAG:
        ChannelIdentifier
            channel_identifier;
};

// Corresponding MHEG datatype: Generic-Numeric
//==========================================================================
enum GenericNumericTag { GENERIC_NUMERIC_CONSTANT_TAG,
GENERIC_NUMERIC_UNSPECIFIED_TAG };
union GenericNumeric
switch (GenericNumericTag){
    case GENERIC_NUMERIC_CONSTANT_TAG:
        long
            constant;
};
```

```
// Interface: RtComponentOrSocket Operation: getSelectionMode
// Corresponding MHEG datatype: none
//==========================================================================
enum SelectionModeValueTag { USER_INTERACTION_TAG, NO_SELECTION_TAG,
MHEG_ACTION_TAG,
USING_APPLICATION_ACTION_TAG };
union SelectionModeValue
switch (SelectionModeValueTag){
    case USER_INTERACTION_TAG:
        unsigned long
            user_interaction;
};

// Interface: RtComponentOrSocket Operation: getModificationMode
// Corresponding MHEG datatype: none
//==========================================================================
enum ModificationModeValueTag { MODIFICATION_MODE_VALUE_USER_INTERACTION_TAG,
MODIFICATION_MODE_VALUE_NO_MODIFICATION_TAG,
MODIFICATION_MODE_VALUE_MHEG_ACTION_TAG,
MODIFICATION_MODE_VALUE_USING_APPLICATION_ACTION_TAG,
MODIFICATION_MODE_VALUE_CHILD_TAG };
union ModificationModeValue
switch (ModificationModeValueTag){
    case MODIFICATION_MODE_VALUE_USER_INTERACTION_TAG:
        unsigned long
            user_interaction;
};

// Corresponding MHEG datatype: External-Identifier
//==========================================================================
enum ExternalIdentifierTag { EXTERNAL_LONG_ID_TAG, PUBLIC_ID_TAG, SYSTEM_ID_TAG
};
union ExternalIdentifier
switch (ExternalIdentifierTag){
    case EXTERNAL_LONG_ID_TAG:
        ExternalLongIdentifier
            external_long_id;
    case PUBLIC_ID_TAG:
        PublicIdentifier
            public_id;
    case SYSTEM_ID_TAG:
        SystemIdentifier
            system_id;
};

// Corresponding MHEG datatype: Container-Tail
//==========================================================================
struct ContainerTail {
    sequence<long>
        indexes;
    enum ContainerTailTag { INDEX_TAG, CONTAINER_CHILD_REF_TAG } tag;
    union ContainerTail
    switch (ContainerTailTag){
        case INDEX_TAG:
            long
                index;
        case CONTAINER_CHILD_REF_TAG:
            ContainerChildReference
                container_child_ref;
      } end;
};

// Corresponding MHEG datatype: Specified-Sizes
//==========================================================================
struct SpecifiedSizes {
    sequence<GenericNumeric,1>
        x_axis_length;
    sequence<GenericNumeric,1>
        y_axis_length;
    sequence<GenericNumeric,1>
        z_axis_length;
};
```

```
// Corresponding MHEG datatype: Attachment-Attribute
//==========================================================================
enum AttachmentAttributeTag { SPECIFIED_POSITION_TAG, LOGICAL_POSITION_TAG };
union AttachmentAttribute
switch (AttachmentAttributeTag){
    case SPECIFIED_POSITION_TAG:
        GenericNumeric
            specified_position;
    case LOGICAL_POSITION_TAG:
        GenericNumeric
            logical_position;
};

// Corresponding MHEG datatype: Length-Attribute
//==========================================================================
enum LengthAttributeTag { SPECIFIED_LENGTH_TAG, RELATIVE_LENGTH_TAG };
union LengthAttribute
switch (LengthAttributeTag){
    case SPECIFIED_LENGTH_TAG:
        GenericNumeric
            specified_length;
    case RELATIVE_LENGTH_TAG:
        GenericNumeric
            relative_length;
};

// Interface: RtComponentOrSocket Operation: getPS
// Interface: RtComponentOrSocket Operation: getPSAP
// Interface: RtComponentOrSocket Operation: getVS
// Interface: RtComponentOrSocket Operation: getVSIAP
// Interface: RtComponentOrSocket Operation: getVSIAPPosition
// Interface: RtComponentOrSocket Operation: getVSEAP
// Interface: RtComponentOrSocket Operation: getVSEAPPosition
// Corresponding MHEG datatype: Specified-Position
//==========================================================================
struct SpecifiedPosition {
    GenericNumeric
        x_point;
    GenericNumeric
        y_point;
    GenericNumeric
        z_point;
};

// Interface: RtComponentOrSocket Operation: setPresentationPriority
// Corresponding MHEG datatype: Presentation-Priority
//==========================================================================
enum PresentationPriorityTag { GENERIC_NUMERIC_TAG, PRIORITY_LEVEL_TAG };
union PresentationPriority
switch (PresentationPriorityTag){
    case GENERIC_NUMERIC_TAG:
        GenericNumeric
            generic_numeric;
    case PRIORITY_LEVEL_TAG:
        PriorityLevel
            priority_level;
};

// Interface: RtComponentOrSocket Operation: setTimestones
// Corresponding MHEG datatype: Timestone
//==========================================================================
struct Timestone {
    long
        timestone_identifier;
    TimestonePosition
        timestone_position;
};
```

```
// Interface: RtComponentOrSocket Operation: setVisibleSize
// Corresponding MHEG datatype: none
//==========================================================================
enum VSTag { X_SIZE_ATTRIBUTE_TAG, Y_SIZE_ATTRIBUTE_TAG, Z_SIZE_ATTRIBUTE_TAG };
union VS
switch (VSTag){
    case X_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            x_size_attribute;
    case Y_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            y_size_attribute;
    case Z_SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            z_size_attribute;
};

// Interface: RtComponentOrSocket Operation: setAttachmentPoint
// Corresponding MHEG datatype: none
//==========================================================================
struct AttachmentPoint {
    sequence<AttachmentAttribute,1>
        x_attachment;
    sequence<AttachmentAttribute,1>
        y_attachment;
    sequence<AttachmentAttribute,1>
        z_attachment;
};

// Interface: RtComponentOrSocket Operation: setAttachmentPointPosition
// Corresponding MHEG datatype: Lengths
//==========================================================================
struct Lengths {
    sequence<LengthAttribute,1>
        x_length;
    sequence<LengthAttribute,1>
        y_length;
    sequence<LengthAttribute,1>
        z_length;
};

// Interface: RtMultiplexedContentOrPresentableSocket Operation: getStreamChosen
// Corresponding MHEG datatype: none
//==========================================================================
enum StreamValueTag { STREAM_IDENTIFIER_TAG, NO_STREAM_CHOSEN_TAG };
union StreamValue
switch (StreamValueTag){
    case STREAM_IDENTIFIER_TAG:
        StreamIdentifier
            stream_identifier;
};

// Interface: MhContent Operation: setData
// Corresponding MHEG datatype: Data-Element
//==========================================================================
struct DataElement {
      sequence<long>
         element_list_index;
      GenericValue
         generic_value;
};

// Interface: NotificationManager Operation: getNotification
// Interface: MhObject Operation: bind
// Interface: MhObject Operation: prepare
// Interface: MhGenericContent Operation: copy
// Corresponding MHEG datatype: Mh-Object-Reference
//==========================================================================
struct MhObjectReference {
enum MhObjectReferenceHeadTag { MHEG_IDENTIFIER_TAG, EXTERNAL_IDENTIFIER_TAG,
ALIAS_TAG, NULL_OBJECT_REF_TAG } head_tag;
union MhObjectReferenceHead
switch (MhObjectReferenceHeadTag){
    case MHEG_IDENTIFIER_TAG:
        MHEGIdentifier
            mheg_identifier;
    case EXTERNAL_IDENTIFIER_TAG:
        ExternalIdentifier
            external_identifier;
    case ALIAS_TAG:
        Alias
            alias;
} head;
enum MhObjectReferenceTailTag { CONTAINER_ELEMENT_REFERENCE_TAG,
OTHER_REFERENCE_TAG
} tail_tag;
union MhObjectReferenceTail
switch (MhObjectReferenceTailTag){
    case CONTAINER_ELEMENT_REFERENCE_TAG:
        ContainerTail
            container_tail;
} tail;
};

// Interface: RtComponentOrSocket Operation: setPerceptibleSizeProjection
// Corresponding MHEG datatype: Perceptible-Size-Projection
//==================================================================
struct PerceptibleSizeProjection {
    sequence<PerceptibleProjection,1>
        x_perceptible_size_projection;
    sequence<PerceptibleProjection,1>
        y_perceptible_size_projection;
    sequence<PerceptibleProjection,1>
        z_perceptible_size_projection;
};

// Corresponding MHEG datatype: Rt-Object-Number-Reference
//==================================================================
enum RtObjectNumberReferenceTag { RT_OBJECT_NUMBER_TAG, RT_DYNAMIC_REFERENCE_TAG
};
union RtObjectNumberReference
switch (RtObjectNumberReferenceTag){
    case RT_OBJECT_NUMBER_TAG:
        long
            rt_object_number;
    case RT_DYNAMIC_REFERENCE_TAG:
        RtDynamicReference
            rt_dynamic_reference;
};

// Interface: RtObject Operation: bind
// Interface: RtObject Operation: new
// Corresponding MHEG datatype: Rt-Object-Reference
//==================================================================
struct RtObjectReference {
    MhObjectReference
        model_object_reference;
    RtObjectNumberReference
        rt_object_number_reference;
};

// Corresponding MHEG datatype: Rt-Reference
//==================================================================
enum RtReferenceTag { RT_REFERENCE_RT_OBJECT_REFERENCE_TAG,
RT_REFERENCE_ALIAS_TAG,
RT_REFERENCE_NULL_RT_OBJECT_TAG };
union RtReference
switch (RtReferenceTag){
    case RT_REFERENCE_RT_OBJECT_REFERENCE_TAG:
        RtObjectReference
            rt_object_reference;
    case RT_REFERENCE_ALIAS_TAG:
        Alias
            alias;
};
```

```
// Corresponding MHEG datatype: Socket-Tail
//=======================================================================
struct SocketTail {
    sequence<long>
        indexes;
    enum SocketTailTag { INDEX_TAG, SUB_SOCKET_REF_TAG } tag;
    union SocketTail
    switch (SocketTailTag){
        case INDEX_TAG:
            long
               index;
        case SUB_SOCKET_REF_TAG:
            SubSocketReference
                sub_socket_ref;
    } end;
};

// Corresponding MHEG datatype: Indexed-Child-Socket
//=======================================================================
struct IndexedChildSocket {
    long
        index;
    SocketTail
        tail;
};

// Interface: Socket Operation: bind
// Interface: Socket Operation: getIdentification
// Corresponding MHEG datatype: Socket-Identification
//=======================================================================
struct SocketIdentification {
    RtReference
        rt_composite_reference;
    SocketTail
        socket_tail;
};

// Interface: Socket Operation: bind
// Corresponding MHEG datatype: Socket-Reference
//=======================================================================
enum SocketReferenceTag { SOCKET_REFERENCE_SOCKET_IDENT_TAG,
SOCKET_REFERENCE_ALIAS_TAG };
union SocketReference
switch (SocketReferenceTag){
    case SOCKET_REFERENCE_SOCKET_IDENT_TAG:
        SocketIdentification
            socket_ident;
    case SOCKET_REFERENCE_ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Rt-Object-Socket-Reference
//=======================================================================
enum RtObjectSocketReferenceTag { RT_REFERENCE_TAG, SOCKET_REFERENCE_TAG };
union RtObjectSocketReference
switch (RtObjectSocketReferenceTag){
    case RT_REFERENCE_TAG:
        RtReference
            rt_reference;
    case SOCKET_REFERENCE_TAG:
        SocketReference
            socket_reference;
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Interface: RtContentOrPresentableSocket Operation: setSliderInteractionStyle
// Interface: RtContentOrPresentableSocket Operation:
//   setEntryFieldInteractionStyle
// Interface: Channel Operation: getAssignedPerceptibles
// Corresponding MHEG datatype: Perceptible-Reference
```

```
//==========================================================================
enum PerceptibleReferenceTag { RT_COMPONENT_REFERENCE_TAG,
RT_SOCKET_REFERENCE_TAG };
union PerceptibleReference
switch (PerceptibleReferenceTag){
    case RT_COMPONENT_REFERENCE_TAG:
        RtReference
            rt_component_reference;
    case RT_SOCKET_REFERENCE_TAG:
        SocketReference
            rt_socket_reference;
};

// Interface: RtCompositeOrStructuralSocket Operation:
//   setScrollingListInteractionStyle
// Corresponding MHEG datatype: Separator
//==========================================================================
enum SeparatorTag { NO_TAG, YES_DEFAULT_TAG, SEPARATOR_PIECE_TAG };
union Separator
switch (SeparatorTag){
    case SEPARATOR_PIECE_TAG:
        PerceptibleReference
            separator_piece;
};

// Interface: RtCompositeOrStructuralSocket Operation: setMenuInteractionStyle
// Corresponding MHEG datatype: Association
//==========================================================================
struct Association {
    sequence<SocketReference,1>
        title;
    sequence<Separator,1>
        separator;
    sequence<SocketReference,1>
        submenu;
    sequence<PresentationPersistence,1>
        submenu_presentation_persistence;
    sequence<Orientation,1>
        submenu_orientation;
};

// Interface: RtSocket Operation: plug
// Corresponding MHEG datatype: Plug-In
//==========================================================================
enum PlugInTag { PLUG_IN_RT_COMPONENT_REFERENCE_TAG,
PLUG_IN_COMPONENT_REFERENCE_TAG,
PLUG_IN_LABEL_TAG };
union PlugIn
switch (PlugInTag){
    case PLUG_IN_RT_COMPONENT_REFERENCE_TAG:
        RtObjectReference
            rt_component_reference;
    case PLUG_IN_COMPONENT_REFERENCE_TAG:
        MhObjectReference
            component_reference;
    case PLUG_IN_LABEL_TAG:
        GenericString
            label;
};

// Interface: RtComponentOrSocket Operation: getVSEAPPosition
// Corresponding MHEG datatype: none
//==========================================================================
enum ReferencePointTag { VSEAP_POSITION_ORIGIN_RGS_TAG,
VSEAP_POSITION_ORIGIN_CGS_TAG,
VSEAP_POSITION_SAME_RGS_COMPONENT_TAG, VSEAP_POSITION_SAME_CGS_COMPONENT_TAG,
VSEAP_POSITION_SPECIFIED_POSITION_TAG };
union ReferencePoint
switch (ReferencePointTag){
    case VSEAP_POSITION_SAME_RGS_COMPONENT_TAG:
        RtObjectSocketReference
            same_RGS_component;
    case VSEAP_POSITION_SAME_CGS_COMPONENT_TAG:
        RtObjectSocketReference
            same_CGS_component;
    case VSEAP_POSITION_SPECIFIED_POSITION_TAG:
        SpecifiedPosition
            specified_position;
};

// Interface: RtScript Operation: setParameters
// Corresponding MHEG datatype: Parameter
//==========================================================================
enum ParameterTag { GENERIC_VALUE_TAG, MH_OBJECT_REFERENCE_TAG };
union Parameter
switch (ParameterTag){
    case GENERIC_VALUE_TAG:
        GenericValue
            generic_value;
    case MH_OBJECT_REFERENCE_TAG:
        MhObjectReference
            mh_object_reference;
};

// Interface: RtObjectOrSocket Operation: setGlobalBehaviour
// Corresponding MHEG datatype: Global-Behaviour
//==========================================================================
enum GlobalBehaviourTag { GLOBAL_BEHAVIOUR_RT_REFERENCE_TAG,
GLOBAL_BEHAVIOUR_GENERIC_LIST_TAG,
GLOBAL_BEHAVIOUR_UNSPECIFIED_TAG };
union GlobalBehaviour
switch (GlobalBehaviourTag){
    case GLOBAL_BEHAVIOUR_RT_REFERENCE_TAG:
        RtReference
            rt_reference;
    case GLOBAL_BEHAVIOUR_GENERIC_LIST_TAG:
        GenericValue
            generic_list;
};

// Interface: RtComponentOrSocket Operation: setVisibleSizesAdjustment
// Corresponding MHEG datatype: Adjustment-PolicyADJUSTMENT-POLICY-
//==========================================================================
enum AdjustmentPolicyTag { ADJUSTMENT_POLICY_COMPONENT_REFERENCE_TAG,
ADJUSTMENT_POLICY_SPECIFIED_TAG, ADJUSTMENT_POLICY_GREATEST_TAG,
ADJUSTMENT_POLICY_SMALLEST_TAG
};
union AdjustmentPolicy
switch (AdjustmentPolicyTag){
    case ADJUSTMENT_POLICY_COMPONENT_REFERENCE_TAG:
        RtObjectSocketReference
            component_reference;
    case ADJUSTMENT_POLICY_SPECIFIED_TAG:
        SpecifiedSizes
            specified;
};

// Interface: RtObject Operation: bind
// Interface: RtObject Operation: new
// Interface: RtObject Operation: getIdentifier
// Corresponding MHEG datatype: none
//==========================================================================
struct RtObjectIdentifier {
    MHEGIdentifier
        model_object_id;
    long
        rt_object_number;
};

// Interface: RtGenericContentOrPresentableSocket Operation:
//   setButtonInteractionStyle
// Corresponding MHEG datatype: Alternate-Presentation-State
//==========================================================================
struct AlternatePresentation {
    PresentationState
        presentation_state;
    PerceptibleReference
        perceptible_target;
};

// Exceptions
//=====================================================================
exception InvalidTarget {
    unsigned short period;
};

enum CompletionStatus { YES, NO};

exception InvalidParameter {
    CompletionStatus completion_status;
    unsigned short period;
};

typedef long EntityIdentifier;

exception AlreadyBound {
    EntityIdentifier entity_identifier;
};

exception NotBound {};

interface MHEGEngine {

    void
        initialiseEngine();
    void
        shutdownEngine();

};

interface NotificationManager {

    sequence<unsigned short>
        getReturnability();

    void
        getNotification(
            in unsigned short
                notification_number,
            out sequence<GenericValue>
                values,
            out sequence<MhObjectReference>
                objects)
    raises(InvalidParameter);

};

interface    EntityManager {

    sequence<MHEGIdentifier>
        getAvailableMhObjects();

    sequence<RtObjectIdentifier>
        getAvailableRtObjects();

    sequence<ChannelIdentifier>
        getAvailableChannels();

    void
        releaseAlias(
            in string
                alias)
    raises(InvalidParameter);

};

interface Entity {

    void
        setAlias(
            in string
                alias)
    raises(InvalidTarget);
    string
        getAlias()
    raises(InvalidTarget);
};

interface MhObject: Entity {

    MHEGIdentifier
        bind(
            in MhObjectReference
                mh_object_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        unbind()
    raises(NotBound);

    MHEGIdentifier
        prepare(
            in MhObjectReference
                mh_object_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        destroy()
    raises(NotBound, InvalidTarget);

    PreparationStatusValue
        getPreparationStatus()
    raises(NotBound, InvalidTarget);

    MHEGIdentifier
        getIdentifier()
    raises(NotBound);

    void
        kill();

};

interface MhAction: MhObject {

    void
        delay(
            in unsigned short
                nested_action_number,
            in unsigned long
                delay)
    raises(InvalidTarget, InvalidParameter);

};

interface MhLink: MhObject {

    void
        abort()
    raises(InvalidTarget);
};

interface MhModel: MhObject {};

interface MhComponent: MhModel {};

interface MhGenericContent: MhComponent {

    void
        copy(
            in sequence<MhObjectReference>
                copies)
    raises(InvalidTarget, InvalidParameter);
};
```

```
interface MhContent: MhGenericContent {

    void
        setData(
            in boolean
                substitution_indicator,
            in sequence<DataElement>
                data_elements)
    raises(InvalidTarget, InvalidParameter);

    GenericValue
        getData(
            in sequence<long>
                element_list_index)
    raises(InvalidTarget, InvalidParameter);

};

interface MhMultiplexedContent: MhGenericContent {

    void
        setMultiplex(
            in sequence<StreamIdentifier>
                stream_list)
    raises(InvalidTarget, InvalidParameter);

    void
        setDemultiplex(
            in sequence<StreamIdentifier>
                stream_list)
    raises(InvalidTarget, InvalidParameter);

};

interface MhComposite: MhComponent {};

interface MhScript: MhModel {};

interface MhContainer: MhObject {};

interface MhDescriptor: MhObject {};

interface RtObjectOrSocket {

    void
        setGlobalBehaviour(
            in GlobalBehaviour
                global_behaviour)
    raises(InvalidTarget, InvalidParameter);

    GenericValue
        getGlobalBehaviour()
    raises(InvalidTarget);

    void
        run()
    raises(InvalidTarget);

    void
        stop()
    raises(InvalidTarget);

};

interface RtObject: Entity {

    RtObjectIdentifier
        bind(
            in RtObjectReference
                rt_object_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        unbind()
    raises(NotBound);
    RtObjectIdentifier
        new(
            in RtObjectReference
                rt_object_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        delete()
    raises(NotBound, InvalidTarget);

    RtAvailabilityStatusValue
        getAvailabilityStatus()
    raises(NotBound, InvalidTarget);

    RtObjectIdentifier
        getIdentifier()
    raises(NotBound);

    void
        kill();

    RunningStatusValue
        getRunningStatus()
    raises(InvalidTarget);

};

interface Socket: Entity, RtObjectOrSocket {

    SocketIdentification
        bind(
            in SocketReference
                socket_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        unbind()
        raises(NotBound);

    SocketIdentification
        getIdentifier();
    void
        kill();

    void
        plug(
            in PlugIn
                plug_in)
    raises(InvalidTarget);

    void
        setVisibleDurationPosition(
            in VisibleDurationPosition
                visible_duration_position)
    raises(InvalidTarget, InvalidParameter);

    unsigned long
        getVisibleDurationPosition()
    raises(InvalidTarget);

};

interface RtScript: RtObject {

    void
        setParameters(
            in sequence<Parameter>
                parameters)
    raises(InvalidTarget);

    TerminationStatusValue
        getTerminationStatus()
    raises(InvalidTarget);

};

interface RtComponentOrSocket {
```

```
void
    setRGS(
        in ChannelIdentifier
            channel_identifier)
raises(InvalidTarget);

RGSValue
    getRGS()
raises(InvalidTarget);

void
    setOpacity(
        in unsigned short
            opacity_rate,
        in unsigned long
            transition_duration)
raises(InvalidTarget);

void
    setPresentationPriority(
        in PresentationPriority
            presentation_priority,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

unsigned short
    getOpacity()
raises(InvalidTarget);

unsigned short
    getEffectiveOpacity()
raises(InvalidTarget);

unsigned short
    getPresentationPriority()
raises(InvalidTarget);

void
    setVisibleDuration(
        in TemporalPosition
            initial_temporal_position,
        in TemporalPosition
            terminal_temporal_position)
raises(InvalidTarget, InvalidParameter);

void
    setTemporalTermination(
        in TemporalTermination
            temporal_termination)
raises(InvalidTarget);

void
    setCurrentTemporalPosition(
        in TemporalPosition
            temporal_position)
raises(InvalidTarget, InvalidParameter);

void
    setSpeed(
        in Speed
            the_speed,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setTimestones(
        in sequence<Timestone>
            timestones)
raises(InvalidTarget, InvalidParameter);

unsigned long
    getInitialTemporalPosition()
raises(InvalidTarget);
```

```
unsigned long
    getTerminalTemporalPosition()
raises(InvalidTarget);

unsigned long
    getVDLength(
        in GTIndicator
            gt_indicator)
raises(InvalidTarget);

TemporalTermination
    getTemporalTermination()
raises(InvalidTarget);

unsigned long
    getCurrentTemporalPosition()
raises(InvalidTarget);

short
    getSpeedRate()
raises(InvalidTarget);

unsigned long
    getOGTR()
raises(InvalidTarget);

short
    getEffectiveSpeedRate()
raises(InvalidTarget);

unsigned long
    getEffectiveOGTR()
raises(InvalidTarget);

unsigned short
    getTimestoneStatus()
raises(InvalidTarget);

void
    setPerceptibleSizeProjection(
        in PerceptibleSizeProjection
            perceptible_size_projection,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setAspectRatio(
        in AspectRatio
            preserved)
raises(InvalidTarget);

void
    setVisibleSize(
        in VSGS
            the_vsgs,
        in VS
            the_vs,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setVisibleSizesAdjustment(
        in sequence<AdjustmentAxis>
            set_of_axes,
        in AdjustmentPolicy
            adjustment_policy,
        in unsigned long
            transition_duration)
raises(InvalidTarget);

void
    setBox(
        in BoxConstants
            box)
raises(InvalidTarget);
```

```
void
    setDefaultBackground(
        in unsigned short
            background,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setAttachmentPoint(
        in AttachmentPointType
            type,
        in AttachmentPoint
            positions)
raises(InvalidTarget, InvalidParameter);

void
    setAttachmentPointPosition(
        in AttachmentPointType
            type,
        in ReferenceType
            vseap_reference_point,
        in Lengths
            the_lengths,
        in unsigned long
            transition_duration)
raises(InvalidTarget, InvalidParameter);

void
    setVisibleSizesAlignment(
        in SizeBorder
            size_border,
        in long
            interval,
        in unsigned long
            transition_duration)
raises(InvalidTarget);

void
    setMovingAbility(
        in UserControls
            moving_ability)
raises(InvalidTarget);

void
    setResizingAbility(
        in UserControls
            resizing_ability)
raises(InvalidTarget);

void
    setScalingAbility(
        in UserControls
            scaling_ability)
raises(InvalidTarget);

void
    setScrollingAbility(
        in UserControls
            scrolling_ability)
raises(InvalidTarget);

unsigned short
    getGSR()
raises(InvalidTarget);

SpecifiedPosition
    getPS(
        in GSIndicator
            gs)
raises(InvalidTarget);

AspectRatio
    getAspectRatio()
raises(InvalidTarget);
```

```
SpecifiedPosition
    getPSAP(
        in PointType
            point_type)
raises(InvalidTarget);

VSGS
    getVSGS()
raises(InvalidTarget);

SpecifiedPosition
    getVS()
raises(InvalidTarget);

BoxConstants
    getBox()
raises(InvalidTarget);

unsigned short
    getDefaultBackground()
raises(InvalidTarget);

SpecifiedPosition
    getVSIAP(
        in PointType
            point_type)
raises(InvalidTarget);

SpecifiedPosition
    getVSIAPPosition()
raises(InvalidTarget);

SpecifiedPosition
    getVSEAP(
        in PointType
            point_type)
raises(InvalidTarget);

SpecifiedPosition
    getVSEAPPosition(
        in ReferencePoint
            reference_point)
raises(InvalidTarget);

UserControls
    getMovingAbility()
raises(InvalidTarget);

UserControls
    getResizingAbility()
raises(InvalidTarget);

UserControls
    getScalingAbility()
raises(InvalidTarget);

UserControls
    getScrollingAbility()
raises(InvalidTarget);

void
    setSelectability(
        in unsigned short
            min_number_of_selections,
        in unsigned short
            max_number_of_selections)
raises(InvalidTarget, InvalidParameter);

void
    setSelectionStatus(
        in SelectionStatusValue
            selection_state)
raises(InvalidTarget);

void
    setSelectionPresentationEffectResponsibility(
        in Responsibility
            the_responsibility)
    raises(InvalidTarget);

    void
        getSelectability(
            out unsigned short
                min_number_of_selections,
            out unsigned short
                max_number_of_selections)
    raises(InvalidTarget);

    EffectiveSelectability
        getEffectiveSelectability()
    raises(InvalidTarget);

    SelectionStatusValue
        getSelectionStatus()
    raises(InvalidTarget);

    SelectionModeValue
        getSelectionMode()
    raises(InvalidTarget);

    Responsibility
        getSelectionPresentationEffectResponsibility()
    raises(InvalidTarget);

    void
        setModifiability(
            in unsigned short
                min_number_of_modifications,
            in unsigned short
                max_number_of_modifications)
    raises(InvalidTarget, InvalidParameter);

    void
        setModificationStatus(
            in ModificationStatusValue
                modification_state)
    raises(InvalidTarget);

    void
        setModificationPresentationEffectResponsibility(
            in Responsibility
                the_responsibility)
    raises(InvalidTarget);

    void
        getModifiability(
            out unsigned short
                min_numbers_of_modifications,
            out unsigned short
                max_numbers_of_modifications)
    raises(InvalidTarget);

    EffectiveModifiability
        getEffectiveModifiability()
    raises(InvalidTarget);

    ModificationStatusValue
        getModificationStatus()
    raises(InvalidTarget);

    ModificationModeValue
        getModificationMode()
    raises(InvalidTarget);

    Responsibility
        getModificationPresentationEffectResponsibility()
    raises(InvalidTarget);

    void
        setNoInteractionStyle()
    raises(InvalidTarget);

};
```

```
interface   RtComponent: RtComponentOrSocket, RtObject {};

interface   RtCompositeOrStructuralSocket {

    void
        setResizingStrategy(
            in ResizingStrategy
                resizing_strategy)
    raises(InvalidTarget);

    ResizingStrategy
        getResizingStrategy()
    raises(InvalidTarget);

    void
        setAudibleCompositionEffect(
            in unsigned short
                audible_effect,
            in unsigned long
                transition_duration)
    raises(InvalidTarget);

    unsigned short
        getAudibleCompositionEffect()
    raises(InvalidTarget);

    unsigned short
        getNumberOfSelectedSockets()
    raises(InvalidTarget);

    unsigned short
        getNumberOfModifiedSockets()
    raises(InvalidTarget);

    void
        setMenuInteractionStyle(
            in Orientation
                upper_menu_orientation,
            in sequence <Association>
                list_of_associations)
    raises(InvalidTarget, InvalidParameter);

    void
        setScrollingListInteractionStyle(
            in PerceptibleReference
                background,
            in unsigned short
                visible_items_number,
            in SocketTail
                first_item,
            in Separator
                the_separator,
            in Orientation
                the_orientation,
            in SliderSide
                slider_side,
            in PerceptibleReference
                slider,
            in PerceptibleReference
                slider_cursor,
            in PerceptibleReference
                slider_background,
            in long
                slider_min_value,
            in long
                slider_max_value)
    raises(InvalidTarget, InvalidParameter);

};
interface RtComposite: RtCompositeOrStructuralSocket, RtComponent {};

interface StructuralSocket: RtCompositeOrStructuralSocket, Socket {};

interface RtGenericContentOrPresentableSocket {
    void
        setAudibleVolume(
            in AudibleVolume
                audible_volume,
            in unsigned long
                transition_duration)
    raises(InvalidTarget, InvalidParameter);

    unsigned long
        getInitialOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getCurrentOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getEffectiveOriginalAudibleVolume()
    raises(InvalidTarget);

    unsigned long
        getPerceptibleAudibleVolume()
    raises(InvalidTarget);

    void
        setButtonInteractionStyle(
            in PresentationState
                initial_state,
            in AlternatePresentation
                alternate_presentation_1,
            in AlternatePresentation
                alternate_presentation_2,
            in AlternatePresentation
                alternate_presentation_3)
    raises(InvalidTarget, InvalidParameter);

};
interface  RtGenericContent: RtGenericContentOrPresentableSocket, RtComponent {};

interface  GenericPresentableSocket: RtGenericContentOrPresentableSocket, Socket
{};

interface RtContentOrPresentableSocket {

    void
        setSliderInteractionStyle(
            in PerceptibleReference
                cursor,
            in PerceptibleReference
                background,
            in Orientation
                the_orientation,
            in short
                min_value,
            in short
                max_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setEntryFieldInteractionStyle(
            in EchoStyle
                echo_style,
            in PerceptibleReference
                background)
    raises(InvalidTarget, InvalidParameter);

};

interface RtContent: RtContentOrPresentableSocket, RtGenericContent {};

interface PresentableSocket: RtContentOrPresentableSocket,
GenericPresentableSocket {};

interface RtMultiplexedContentOrPresentableSocket {
    void
        setStreamChoice(
            in StreamIdentifier
                stream_identifier)
    raises(InvalidTarget, InvalidParameter);

    StreamValue
        getStreamChosen()
    raises(InvalidTarget);
};

interface RtMultiplexedContent: RtMultiplexedContentOrPresentableSocket,
RtGenericContent {};

interface MultiplexedPresentableSocket:  RtMultiplexedContentOrPresentableSocket,
GenericPresentableSocket {};

interface Channel: Entity {

    ChannelIdentifier
        bind(
            in ChannelReference
                channel_reference)
    raises(AlreadyBound, InvalidTarget);

    void
        unbind()
    raises(NotBound);

    ChannelIdentifier
        new(
            in ChannelReference
                channel_reference,
            in OriginalDefDeclaration
                original_definition_declaration)
    raises(AlreadyBound, InvalidTarget);

    void
        delete()
    raises(NotBound, InvalidTarget);

    ChannelStatusValue
        getAvailability()
    raises(NotBound, InvalidTarget);

    ChannelIdentifier
        getIdentifier()
    raises(NotBound);

    void
        kill();

    void
        setPerceptability(
            in ChannelPerceptabilityValue
                channel_perceptability)
    raises(InvalidTarget);

    ChannelPerceptabilityValue
        getPerceptability()
    raises(InvalidTarget);

    sequence<PerceptibleReference>
        getAssignedPerceptibles()
    raises(InvalidTarget);

};

module x {

interface EvaluatedValue;

// Corresponding MHEG datatype: RGS-Constants
//==========================================================
enum RGSConstants {
    NULL_CHANNEL,
    PRGS
};
```

```
// Corresponding MHEG datatype: Stream-Chosen
//===========================================================================
enum StreamChosen {
    NO_STREAM_CHOSEN
};

// Corresponding MHEG datatype: Selection-Mode-Constants
//===========================================================================
enum SelectionModeConstants {
    NO_SELECTION,
    MHEG_ACTION,
    USING_APPLICATION_ACTION
};

// Corresponding MHEG datatype: Modification-Mode-Constants
//===========================================================================
enum ModificationModeConstants {
    MODIFICATION_MODE_CONSTANTS_NO_MODIFICATION,
    MODIFICATION_MODE_CONSTANTS_MHEG_ACTION,
    MODIFICATION_MODE_CONSTANTS_USING_APPLICATION_ACTION,
    MODIFICATION_MODE_CONSTANTS_CHILD
};

// Corresponding MHEG datatype: Comparison-Value-Constant
//===========================================================================
enum ComparisonValueConstantTag { TEMPORAL_TERMINATION_TAG,
RESIZING_STRATEGY_TAG, VSGS_TAG, RESPONSIBILITY_TAG,
PREPARATION_STATUS_VALUE_TAG, RT_AVAILABILITY_STATUS_VALUE_TAG,
RUNNING_STATUS_VALUE_TAG, TERMINATION_STATUS_VALUE_TAG, RGS_CONSTANTS_TAG,
USER_CONTROLS_TAG, ASPECT_RATIO_TAG, BOX_CONSTANTS_TAG, STREAM_CHOSEN_TAG,
EFFECTIVE_SELECTABILITY_TAG, SELECTION_STATUS_VALUE_TAG,
SELECTION_MODE_CONSTANTS_TAG, EFFECTIVE_MODIFIABILITY_TAG,
MODIFICATION_STATUS_VALUE_TAG, MODIFICATION_MODE_CONSTANTS_TAG,
CHANNEL_STATUS_VALUE_TAG, CHANNEL_PERCEPTABILITY_VALUES_TAG };
union ComparisonValueConstant
switch (ComparisonValueConstantTag){
    case TEMPORAL_TERMINATION_TAG:
        TemporalTermination
            temporal_termination;
    case RESIZING_STRATEGY_TAG:
        ResizingStrategy
            resizing_strategy;
    case VSGS_TAG:
        VSGS
            vsgs;
    case RESPONSIBILITY_TAG:
        Responsibility
            responsibility;
    case PREPARATION_STATUS_VALUE_TAG:
        PreparationStatusValue
            preparation_status_value;
    case RT_AVAILABILITY_STATUS_VALUE_TAG:
        RtAvailabilityStatusValue
            rt_availability_status_value;
    case RUNNING_STATUS_VALUE_TAG:
        RunningStatusValue
            running_status_value;
    case TERMINATION_STATUS_VALUE_TAG:
        TerminationStatusValue
            termination_status_value;
    case RGS_CONSTANTS_TAG:
        RGSConstants
            rgs_constants;
    case USER_CONTROLS_TAG:
        UserControls
            user_controls;
    case ASPECT_RATIO_TAG:
        AspectRatio
            aspect_ratio;
    case BOX_CONSTANTS_TAG:
        BoxConstants
            box_constants;
    case STREAM_CHOSEN_TAG:
        StreamChosen
            stream_chosen;
    case EFFECTIVE_SELECTABILITY_TAG:
        EffectiveSelectability
            effective_selectability;
    case SELECTION_STATUS_VALUE_TAG:
        SelectionStatusValue
            selection_status_value;
    case SELECTION_MODE_CONSTANTS_TAG:
        SelectionModeConstants
            selection_mode_constants;
    case EFFECTIVE_MODIFIABILITY_TAG:
        EffectiveModifiability
            effective_modifiability;
    case MODIFICATION_STATUS_VALUE_TAG:
        ModificationStatusValue
            modification_status_value;
    case MODIFICATION_MODE_CONSTANTS_TAG:
        ModificationModeConstants
            modification_mode_constants;
    case CHANNEL_STATUS_VALUE_TAG:
        ChannelStatusValue
            channel_status_value;
    case CHANNEL_PERCEPTABILITY_VALUES_TAG:
        ChannelPerceptabilityValue
            channel_perceptability_value;
};

// Corresponding MHEG datatype: Data-Reference
//===========================================================================
enum DataReferenceTag { EXTERNAL_IDENTIFIER_TAG,ALIAS_TAG };
union DataReference
switch (DataReferenceTag){
    case EXTERNAL_IDENTIFIER_TAG:
        ExternalIdentifier
            external_identifier;
    case ALIAS_TAG:
        Alias
            alias;
};

// Corresponding MHEG datatype: Reference
//===========================================================================
enum ReferenceTag { MHEG_OBJECT_REF_TAG, RT_OBJECT_REF_TAG, CHANNEL_ID_TAG,
SOCKET_IDENT_REF_TAG, NULL_REF_TAG };
union Reference
switch (ReferenceTag){
    case MHEG_OBJECT_REF_TAG:
        MhObjectReference
            mh_object_ref;
    case RT_OBJECT_REF_TAG:
        RtObjectReference
            rt_object_ref;
    case CHANNEL_ID_TAG:
        ChannelIdentifier
            channel_id;
    case SOCKET_IDENT_REF_TAG:
        SocketIdentification
            socket_ident_ref;
};

// Corresponding MHEG datatype: Generic-Reference
//===========================================================================
enum GenericReferenceTag { REFERENCE_TAG, EVALUATED_VALUE_TAG, UNSPECIFIED_TAG };
union GenericReference
switch (GenericReferenceTag){
    case REFERENCE_TAG:
        Reference
            reference;
    case EVALUATED_VALUE_TAG:
        EvaluatedValue
            evaluated_value;
};
```

```
// Corresponding MHEG datatype: Macro-Def-Id
//============================================================================
enum MacroDefIdTag { STRING_FIELD_TAG, NUMERIC_TAG };
union MacroDefId
switch (MacroDefIdTag){
    case STRING_FIELD_TAG:
        string
            string_field;
    case NUMERIC_TAG:
        long
            numeric;
};

// Corresponding MHEG datatype: Target-Macro
//============================================================================
struct TargetMacro {
    MacroDefId
        macro_def_id;
    sequence<GenericReference,1>
        generic_reference;
};

// Corresponding MHEG datatype: Target-Parameter
//============================================================================
enum TargetParameterTag { GENERIC_REFERENCE_TAG, TARGET_MACRO_TAG };
union TargetParameter
switch (TargetParameterTag){
    case GENERIC_REFERENCE_TAG:
        GenericReference
            generic_reference;
    case TARGET_MACRO_TAG:
        TargetMacro
            target_macro;
};

// Corresponding MHEG datatype: Targets-Parameter
//============================================================================
typedef sequence<TargetParameter> TargetsParameter;

// Corresponding MHEG datatype: Mh-Target-Macro
//============================================================================
struct MhTargetMacro {
    MacroDefId
        macro_def_id;
    sequence<MhObjectReference,1>
        mh_object_reference;
};

// Corresponding MHEG datatype: Mh-Target-Parameter
//============================================================================
enum MhTargetParameterTag { MH_REFERENCE_TAG, EVALUATED_TARGET_TAG,
MH_TARGET_MACRO_TAG };
union MhTargetParameter
switch (MhTargetParameterTag){
    case MH_REFERENCE_TAG:
        MhObjectReference
            mh_reference;
    case EVALUATED_TARGET_TAG:
        EvaluatedValue
            evaluated_target;
    case MH_TARGET_MACRO_TAG:
        MhTargetMacro
            mh_target_macro;
};

// Corresponding MHEG datatype: Mh-Targets-Parameter
//============================================================================
typedef sequence<MhTargetParameter> MhTargetsParameter;
```

```
// Corresponding MHEG datatype: Rt-Target-Macro
//=========================================================================
struct RtTargetMacro {
    MacroDefId
        macro_def_id;
    sequence<RtReference,1>
        rt_reference;
};

// Corresponding MHEG datatype: Rt-Target-Parameter
//=========================================================================
enum RtTargetParameterTag { RT_TARGET_PARAMETER_RT_REFERENCE_TAG,
RT_TARGET_PARAMETER_EVALUATED_TARGET_TAG, RT_TARGET_PARAMETER_RT_TARGET_MACRO_TAG
};
union RtTargetParameter
switch (RtTargetParameterTag){
    case RT_TARGET_PARAMETER_RT_REFERENCE_TAG:
        RtReference
            rt_reference;
    case RT_TARGET_PARAMETER_EVALUATED_TARGET_TAG:
        EvaluatedValue
            evaluated_target;
    case RT_TARGET_PARAMETER_RT_TARGET_MACRO_TAG:
        RtTargetMacro
            rt_target_macro;
};

// Corresponding MHEG datatype: Rt-Targets-Parameter
//=========================================================================
typedef sequence<RtTargetParameter> RtTargetsParameter;

// Corresponding MHEG datatype: Socket-Target-Macro
//=========================================================================
struct SocketTargetMacro {
    MacroDefId
        macro_def_id;
    sequence<SocketReference,1>
        socket_reference;
};

// Corresponding MHEG datatype: Socket-Target-Parameter
//=========================================================================
enum SocketTargetParameterTag { SOCKET_TARGET_PARAMETER_SOCKET_REFERENCE_TAG,
SOCKET_TARGET_PARAMETER_EVALUATED_TARGET_TAG,
SOCKET_TARGET_PARAMETER_SOCKET_TARGET_MACRO_TAG };
union SocketTargetParameter
switch (SocketTargetParameterTag){
    case SOCKET_TARGET_PARAMETER_SOCKET_REFERENCE_TAG:
        SocketReference
            socket_reference;
    case SOCKET_TARGET_PARAMETER_EVALUATED_TARGET_TAG:
        EvaluatedValue
            evaluated_target;
    case SOCKET_TARGET_PARAMETER_SOCKET_TARGET_MACRO_TAG:
        SocketTargetMacro
            socket_target_macro;
};

// Corresponding MHEG datatype: Socket-Targets-Parameter
//=========================================================================
typedef sequence<SocketTargetParameter> SocketTargetsParameter;

// Corresponding MHEG datatype: Rt-Socket-Target-Macro
//=========================================================================
struct RtSocketTargetMacro {
    MacroDefId
        macro_def_id;
    sequence<RtObjectSocketReference,1>
        rt_socket_reference;
};
```

```
// Corresponding MHEG datatype: Rt-Socket-Target-Parameter
//=========================================================================
enum RtSocketTargetParameterTag {
RT_SOCKET_TARGET_PARAMETER_RT_SOCKET_REFERENCE_TAG,
RT_SOCKET_TARGET_PARAMETER_EVALUATED_TARGET_TAG,
RT_SOCKET_TARGET_PARAMETER_RT_SOCKET_TARGET_MACRO_TAG };
union RtSocketTargetParameter
switch (RtSocketTargetParameterTag){
    case RT_SOCKET_TARGET_PARAMETER_RT_SOCKET_REFERENCE_TAG:
        RtObjectSocketReference
            rt_socket_reference;
    case RT_SOCKET_TARGET_PARAMETER_EVALUATED_TARGET_TAG:
        EvaluatedValue
            evaluated_target;
    case RT_SOCKET_TARGET_PARAMETER_RT_SOCKET_TARGET_MACRO_TAG:
        RtSocketTargetMacro
            rt_socket_target_macro;
};

// Corresponding MHEG datatype: Rt-Socket-Targets-Parameter
//=========================================================================
typedef sequence<RtSocketTargetParameter> RtSocketTargetsParameter;

// Corresponding MHEG datatype: Channel-Target-Macro
//=========================================================================
struct ChannelTargetMacro {
    MacroDefId
        macro_def_id;
    sequence<ChannelReference,1>
        channel_reference;
};

// Corresponding MHEG datatype: Channel-Target-Parameter
//=========================================================================
enum ChannelTargetParameterTag { CHANNEL_TARGET_PARAMETER_CHANNEL_REFERENCE_TAG,
CHANNEL_TARGET_PARAMETER_EVALUATED_TARGET_TAG,
CHANNEL_TARGET_PARAMETER_CHANNEL_TARGET_MACRO_TAG };
union ChannelTargetParameter
switch (ChannelTargetParameterTag){
    case CHANNEL_TARGET_PARAMETER_CHANNEL_REFERENCE_TAG:
        ChannelReference
            channel_reference;
    case CHANNEL_TARGET_PARAMETER_EVALUATED_TARGET_TAG:
        EvaluatedValue
            evaluated_target;
    case CHANNEL_TARGET_PARAMETER_CHANNEL_TARGET_MACRO_TAG:
        ChannelTargetMacro
            channel_target_macro;
};

// Corresponding MHEG datatype: Channel-Targets-Parameter
//=========================================================================
typedef sequence<ChannelTargetParameter> ChannelTargetsParameter;

// Corresponding MHEG datatype: Element-List-Index-Macro
//=========================================================================
struct ElementListIndexMacro {
    MacroDefId
        macro_def_id;
    sequence<sequence<GenericNumeric>,1>
        generic_numeric_list;
};

// Corresponding MHEG datatype: Element-List-Index-Parameter
//=========================================================================
enum ElementListIndexParameterTag { GENERIC_NUMERIC_LIST_TAG,
ELEMENT_LIST_INDEX_MACRO_TAG };
union ElementListIndexParameter
switch (ElementListIndexParameterTag){
    case GENERIC_NUMERIC_LIST_TAG:
        sequence<GenericNumeric>
            generic_numeric_list;
    case ELEMENT_LIST_INDEX_MACRO_TAG:
        ElementListIndexMacro
            element_list_index_macro;
};

// Corresponding MHEG datatype: Get-Data
//==========================================================================
struct GetData {
    MhTargetParameter
        content_target_parameter;
    sequence<ElementListIndexParameter,1>
        element_list_index_parameter;
};

// Corresponding MHEG datatype: GT-Indicator-Macro
//==========================================================================
struct GTIndicatorMacro {
    MacroDefId
        macro_def_id;
    sequence<GTIndicator,1>
        gt_indicator;
};

// Corresponding MHEG datatype: GT-Indicator-Parameter
//==========================================================================
enum GTIndicatorParameterTag { GT_INDICATOR_TAG, GT_INDICATOR_MACRO_TAG };
union GTIndicatorParameter
switch (GTIndicatorParameterTag){
    case GT_INDICATOR_TAG:
        GTIndicator
            gt_indicator;
    case GT_INDICATOR_MACRO_TAG:
        GTIndicatorMacro
            gt_indicator_macro;
};

// Corresponding MHEG datatype: Get-PD
//==========================================================================
struct GetPD {
    RtSocketTargetParameter
        target;
    sequence<GTIndicatorParameter,1>
        gt_indicator;
};

// Corresponding MHEG datatype: Get-VD-Length
//==========================================================================
struct GetVDLength {
    RtSocketTargetParameter
        target;
    GTIndicatorParameter
        gt_indicator;
};

// Corresponding MHEG datatype: GS-Indicator-Macro
//==========================================================================
struct GSIndicatorMacro {
    MacroDefId
        macro_def_id;
    sequence<GSIndicator,1>
        gs_indicator;
};

// Corresponding MHEG datatype: GS-Indicator-Parameter
//==========================================================================
enum GSIndicatorParameterTag { GS_INDICATOR_TAG, GS_INDICATOR_MACRO_TAG };
union GSIndicatorParameter
switch (GSIndicatorParameterTag){
    case GS_INDICATOR_TAG:
        GSIndicator
            gs_indicator;
    case GS_INDICATOR_MACRO_TAG:
        GSIndicatorMacro
            gs_indicator_macro;
};
```

```
// Corresponding MHEG datatype: Perceptible-Size-Parameter
//==========================================================================
struct PerceptibleSizeParameter {
    RtSocketTargetParameter
        target;
    sequence<GSIndicatorParameter,1>
        generic_space;
};

// Corresponding MHEG datatype: Point-Type-Macro
//==========================================================================
struct PointTypeMacro {
    MacroDefId
        macro_def_id;
    sequence<PointType,1>
        point_type;
};

// Corresponding MHEG datatype: Point-Type-Parameter
//==========================================================================
enum PointTypeParameterTag { POINT_TYPE_TAG, POINT_TYPE_MACRO_TAG };
union PointTypeParameter
switch (PointTypeParameterTag){
    case POINT_TYPE_TAG:
        PointType
            point_type;
    case POINT_TYPE_MACRO_TAG:
        PointTypeMacro
            point_type_macro;
};


// Corresponding MHEG datatype: AP-Parameter
//==========================================================================
struct APParameter {
      RtSocketTargetParameter
         target;
      sequence<PointTypeParameter,1>
         type;
};


// Corresponding MHEG datatype: Reference-Point-Macro
//==========================================================================
struct ReferencePointMacro {
      MacroDefId
         macro_def_id;
      sequence<ReferencePoint,1>
         reference_point;
};


// Corresponding MHEG datatype: Reference-Point-Parameter
//==========================================================================
enum ReferencePointParameterTag { REFERENCE_POINT_TAG, REFERENCE_POINT_MACRO_TAG
};
union ReferencePointParameter
switch (ReferencePointParameterTag){
      case REFERENCE_POINT_TAG:
         ReferencePoint
            reference_point;
      case REFERENCE_POINT_MACRO_TAG:
         ReferencePointMacro
            reference_point_macro;
};
```

```
// Corresponding MHEG datatype: VSEAP-Position-Parameter
//==========================================================================
struct VSEAPPositionParameter {
       RtSocketTargetParameter
          target;
       sequence<ReferencePointParameter,1>
          reference_point;
};


// Corresponding MHEG datatype: Generic-Value
//==========================================================================
enum GenericValueTag { GENERIC_VALUE_BOOLEAN_FIELD_TAG,
GENERIC_VALUE_NUMERIC_TAG, GENERIC_VALUE_STRING_FIELD_TAG,
GENERIC_VALUE_GENERIC_LIST_TAG, GENERIC_VALUE_REFERENCE_TAG,
GENERIC_VALUE_UNSPECIFIED_TAG, GENERIC_VALUE_EVALUATED_VALUE_TAG };
union GenericValue
switch (GenericValueTag){
       case GENERIC_VALUE_BOOLEAN_FIELD_TAG:
          boolean
             boolean_field;
       case GENERIC_VALUE_NUMERIC_TAG:
          long
             numeric;
       case GENERIC_VALUE_STRING_FIELD_TAG:
          string
             string_field;
       case GENERIC_VALUE_GENERIC_LIST_TAG:
          sequence<GenericValue>
             generic_list;
       case GENERIC_VALUE_REFERENCE_TAG:
          Reference
             reference;
       case GENERIC_VALUE_EVALUATED_VALUE_TAG:
          EvaluatedValue
             evaluated_value;
};

// Corresponding MHEG datatype: Generic-Boolean
//==========================================================================
enum GenericBooleanTag { GENERIC_BOOLEAN_CONSTANT_TAG,
GENERIC_BOOLEAN_EVALUATED_VALUE_TAG, GENERIC_BOOLEAN_UNSPECIFIED_TAG };
union GenericBoolean
switch (GenericBooleanTag){
    case GENERIC_BOOLEAN_CONSTANT_TAG:
        boolean
            constant;
    case GENERIC_BOOLEAN_EVALUATED_VALUE_TAG:
        EvaluatedValue
            evaluated_value;
};

// Corresponding MHEG datatype: Generic-Numeric
//==========================================================================
enum GenericNumericTag { GENERIC_NUMERIC_CONSTANT_TAG,
GENERIC_NUMERIC_EVALUATED_VALUE_TAG, GENERIC_NUMERIC_UNSPECIFIED_TAG };
union GenericNumeric
switch (GenericNumericTag){
    case GENERIC_NUMERIC_CONSTANT_TAG:
        long
            constant;
    case GENERIC_NUMERIC_EVALUATED_VALUE_TAG:
        EvaluatedValue
            evaluated_value;
};

// Corresponding MHEG datatype: Generic-String
//==========================================================================
enum GenericStringTag { GENERIC_STRING_CONSTANT_TAG,
GENERIC_STRING_EVALUATED_VALUE_TAG, GENERIC_STRING_UNSPECIFIED_TAG };
union GenericString
switch (GenericStringTag){
    case GENERIC_STRING_CONSTANT_TAG:
        string
            constant;
    case GENERIC_STRING_EVALUATED_VALUE_TAG:
        EvaluatedValue
            evaluated_value;
};

// Corresponding MHEG datatype: Value-Macro
//===========================================================================
struct ValueMacro {
    MacroDefId
        macro_def_id;
    sequence<GenericValue,1>
        generic_value;
};

// Corresponding MHEG datatype: Boolean-Macro
//===========================================================================
struct BooleanMacro {
    MacroDefId
        macro_def_id;
    sequence<GenericBoolean,1>
        generic_boolean;
};

// Corresponding MHEG datatype: Numeric-Macro
//===========================================================================
struct NumericMacro {
    MacroDefId
        macro_def_id;
    sequence<GenericNumeric,1>
        generic_numeric;
};

// Corresponding MHEG datatype: String-Macro
//===========================================================================
struct StringMacro {
    MacroDefId
        macro_def_id;
    sequence<GenericString,1>
        generic_string;
};

// Corresponding MHEG datatype: Boolean-Parameter
//===========================================================================
enum BooleanParameterTag { GENERIC_BOOLEAN_TAG, BOOLEAN_MACRO_TAG };
union BooleanParameter
switch (BooleanParameterTag){
    case GENERIC_BOOLEAN_TAG:
        GenericBoolean
            generic_boolean;
    case BOOLEAN_MACRO_TAG:
        BooleanMacro
            boolean_macro;
};

// Corresponding MHEG datatype: Numeric-Parameter
//===========================================================================
enum NumericParameterTag { GENERIC_NUMERIC_TAG, NUMERIC_MACRO_TAG };
union NumericParameter
switch (NumericParameterTag){
    case GENERIC_NUMERIC_TAG:
        GenericNumeric
            generic_numeric;
    case NUMERIC_MACRO_TAG:
        NumericMacro
            numeric_macro;
};
```

```
// Corresponding MHEG datatype: String-Parameter
//==========================================================================
enum StringParameterTag { GENERIC_STRING_TAG, STRING_MACRO_TAG };
union StringParameter
switch (StringParameterTag){
    case GENERIC_STRING_TAG:
        GenericString
            generic_string;
    case STRING_MACRO_TAG:
        StringMacro
            string_macro;
};

// Corresponding MHEG datatype: Value-Parameter
//==========================================================================
enum ValueParameterTag { GENERIC_VALUE_TAG, VALUE_MACRO_TAG };
union ValueParameter
switch (ValueParameterTag){
    case GENERIC_VALUE_TAG:
        GenericValue
            generic_value;
    case VALUE_MACRO_TAG:
        ValueMacro
            value_macro;
};

// Corresponding MHEG datatype: Generic-Volume-Range
//==========================================================================
struct GenericVolumeRange {
    sequence<long,1>
        maximum_volume;
    sequence<long,1>
        minimum_volume;
};

// Corresponding MHEG datatype: Original-Size
//==========================================================================
struct OriginalSize {
    sequence<long,1>
        x_length;
    sequence<long,1>
        y_length;
    sequence<long,1>
        z_length;
};

// Corresponding MHEG datatype: Original-Perception
//==========================================================================
struct OriginalPerception {
    sequence<long,1>
        initial_GTR;
    sequence<long,1>
        original_duration;
    sequence<OriginalSize,1>
        original_size;
    sequence<GenericVolumeRange,1>
        audible_volume_range;
    sequence<long,1>
        original_volume;
};

// Corresponding MHEG datatype: Content-Encoding-Identification
//==========================================================================
enum ContentEncodingIdentificationTag { MHEG_CONTENT_CATALOGUE_TAG,
PROPRIETARY_CONTENT_CATALOGUE_TAG };
union ContentEncodingIdentification
switch (ContentEncodingIdentificationTag){
    case MHEG_CONTENT_CATALOGUE_TAG:
        long
            mheg_content_catalogue;
    case PROPRIETARY_CONTENT_CATALOGUE_TAG:
        long
            proprietary_content_catalogue;
};
```

```
// Corresponding MHEG datatype: Content-Hook
//==========================================================================
struct ContentHook {
    ContentEncodingIdentification
        content_encoding_identification;
    string
        content_encoding_description;
};

// Corresponding MHEG datatype: Script-Encoding-Identification
//==========================================================================
enum ScriptEncodingIdentificationTag { MHEG_SCRIPT_CATALOGUE_TAG,
PROPRIETARY_SCRIPT_CATALOGUE_TAG };
union ScriptEncodingIdentification
switch (ScriptEncodingIdentificationTag){
    case MHEG_SCRIPT_CATALOGUE_TAG:
        long
            mheg_script_catalogue;
    case PROPRIETARY_SCRIPT_CATALOGUE_TAG:
        long
            proprietary_script_catalogue;
};

// Corresponding MHEG datatype: Script-Hook
//==========================================================================
struct ScriptHook {
    ScriptEncodingIdentification
        script_encoding_identification;
    string
        script_encoding_description;
};

// Corresponding MHEG datatype: Content-Classification
//==========================================================================
enum ContentClassificationTag { MHEG_CONTENT_CLASSIFICATION_TAG,
PROPRIETARY_CONTENT_CLASSIFICATION_TAG };
union ContentClassification
switch (ContentClassificationTag){
    case MHEG_CONTENT_CLASSIFICATION_TAG:
        long
            mheg_content_classification;
    case PROPRIETARY_CONTENT_CLASSIFICATION_TAG:
        long
            proprietary_content_classification;
};

// Corresponding MHEG datatype: Script-Classification
//==========================================================================
enum ScriptClassificationTag { MHEG_SCRIPT_CLASSIFICATION_TAG,
PROPRIETARY_SCRIPT_CLASSIFICATION_TAG };
union ScriptClassification
switch (ScriptClassificationTag){
    case MHEG_SCRIPT_CLASSIFICATION_TAG:
        long
            mheg_script_classification;
    case PROPRIETARY_SCRIPT_CLASSIFICATION_TAG:
        long
            proprietary_script_classification;
};

// Corresponding MHEG datatype: Evaluated-Value
//==========================================================================
interface EvaluatedValue {
enum EvaluatedValueTag { GET_PREPARATION_STATUS_TAG, GET_DATA_TAG,
GET_RT_AVAILABILITY_STATUS_TAG, GET_GLOBAL_BEHAVIOUR_TAG, GET_RUNNING_STATUS_TAG,
GET_TERMINATION_STATUS_TAG, GET_RGS_TAG, GET_OPACITY_TAG,
GET_EFFECTIVE_OPACITY_TAG, GET_PRESENTATION_PRIORITY_TAG, GET_PD_TAG,
GET_INITIAL_TEMPORAL_POSITION_TAG, GET_TERMINAL_TEMPORAL_POSITION_TAG,
GET_VD_LENGTH_TAG, GET_CURRENT_TEMPORAL_POSITION_TAG, GET_VD_POSITION_TAG,
GET_SPEED_RATE_TAG, GET_OGTR_TAG, GET_EFFECTIVE_SPEED_RATE_TAG,
GET_EFFECTIVE_OGTR_TAG, GET_TIMESTONE_STATUS_TAG, GET_GSR_TAG, GET_PS_TAG,
GET_RESIZING_STRATEGY_TAG, GET_ASPECT_RATIO_TAG, GET_PSAP_TAG, GET_VSGS_TAG,
GET_VS_TAG, GET_BOX_TAG, GET_DEFAULT_BACKGROUND_TAG, GET_VSIAP_TAG,
GET_VSIAP_POSITION_TAG, GET_VSEAP_TAG, GET_VSEAP_POSITION_TAG,
GET_MOVING_ABILITY_TAG, GET_RESIZING_ABILITY_TAG, GET_SCALING_ABILITY_TAG,
GET_SCROLLING_ABILITY_TAG, GET_IOV_TAG, GET_CURRENT_OV_TAG,
GET_AUDIBLE_COMPOSITION_EFFECT_TAG, GET_EFFECTIVE_OV_TAG, GET_PERCEPTIBLE_OV_TAG,
GET_STREAM_CHOSEN_TAG, GET_SELECTABILITY_TAG, GET_EFFECTIVE_SELECTABILITY_TAG,
GET_SELECTION_STATUS_TAG, GET_SELECTION_MODE_TAG,
GET_NUMBER_SELECTED_SOCKETS_TAG,
GET_SELECTION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG, GET_MODIFIABILITY_TAG,
GET_EFFECTIVE_MODIFIABILITY_TAG, GET_MODIFICATION_STATUS_TAG,
GET_MODIFICATION_MODE_TAG, GET_NUMBER_MODIFIED_SOCKETS_TAG,
GET_MODIFICATION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG,
GET_CHANNEL_AVAILABILITY_STATUS_TAG, GET_CHANNEL_PERCEPTABILITY_TAG };
union EvaluatedValue
switch (EvaluatedValueTag){
    case GET_PREPARATION_STATUS_TAG:
        MhTargetParameter
            get_preparation_status;
    case GET_DATA_TAG:
        GetData
            get_data;
    case GET_RT_AVAILABILITY_STATUS_TAG:
        RtTargetParameter
            get_rt_availability_status;
    case GET_GLOBAL_BEHAVIOUR_TAG:
        RtTargetParameter
            get_global_behaviour;
    case GET_RUNNING_STATUS_TAG:
        RtTargetParameter
            get_running_status;
    case GET_TERMINATION_STATUS_TAG:
        RtTargetParameter
            get_termination_status;
    case GET_RGS_TAG:
        RtSocketTargetParameter
            get_RGS;
    case GET_OPACITY_TAG:
        RtSocketTargetParameter
            get_opacity;
    case GET_EFFECTIVE_OPACITY_TAG:
        RtSocketTargetParameter
            get_effective_opacity;
    case GET_PRESENTATION_PRIORITY_TAG:
        RtSocketTargetParameter
            get_presentation_priority;
    case GET_PD_TAG:
        GetPD
            get_PD;
    case GET_INITIAL_TEMPORAL_POSITION_TAG:
        RtSocketTargetParameter
            get_initial_temporal_position;
    case GET_TERMINAL_TEMPORAL_POSITION_TAG:
        RtSocketTargetParameter
            get_terminal_temporal_position;
    case GET_VD_LENGTH_TAG:
        GetVDLength
            get_VD_length;
    case GET_CURRENT_TEMPORAL_POSITION_TAG:
        RtSocketTargetParameter
            get_current_temporal_position;
    case GET_VD_POSITION_TAG:
        SocketTargetParameter
            get_VD_position;
    case GET_SPEED_RATE_TAG:
        RtSocketTargetParameter
            get_speed_rate;
```

```
case GET_OGTR_TAG:
    RtSocketTargetParameter
        get_OGTR;
case GET_EFFECTIVE_SPEED_RATE_TAG:
    RtSocketTargetParameter
        get_effective_speed_rate;
case GET_EFFECTIVE_OGTR_TAG:
    RtSocketTargetParameter
        get_effective_OGTR;
case GET_TIMESTONE_STATUS_TAG:
    RtSocketTargetParameter
        get_timestone_status;
case GET_GSR_TAG:
    RtSocketTargetParameter
        get_GSR;
case GET_PS_TAG:
    PerceptibleSizeParameter
        get_PS;
case GET_RESIZING_STRATEGY_TAG:
    RtSocketTargetParameter
        get_resizing_strategy;
case GET_ASPECT_RATIO_TAG:
    RtSocketTargetParameter
        get_aspect_ratio;
case GET_PSAP_TAG:
    APParameter
        get_PSAP;
case GET_VSGS_TAG:
    RtSocketTargetParameter
        get_VSGS;
case GET_VS_TAG:
    RtSocketTargetParameter
        get_VS;
case GET_BOX_TAG:
    RtSocketTargetParameter
        get_box;
case GET_DEFAULT_BACKGROUND_TAG:
    RtSocketTargetParameter
        get_default_background;
case GET_VSIAP_TAG:
    APParameter
        get_VSIAP;
case GET_VSIAP_POSITION_TAG:
    RtSocketTargetParameter
        get_VSIAP_position;
case GET_VSEAP_TAG:
    APParameter
        get_VSEAP;
case GET_VSEAP_POSITION_TAG:
    VSEAPPositionParameter
        get_VSEAP_position;
case GET_MOVING_ABILITY_TAG:
    RtSocketTargetParameter
        get_moving_ability;
case GET_RESIZING_ABILITY_TAG:
    RtSocketTargetParameter
        get_resizing_ability;
case GET_SCALING_ABILITY_TAG:
    RtSocketTargetParameter
        get_scaling_ability;
case GET_SCROLLING_ABILITY_TAG:
    RtSocketTargetParameter
        get_scrolling_ability;
case GET_IOV_TAG:
    RtSocketTargetParameter
        get_IOV;
case GET_CURRENT_OV_TAG:
    RtSocketTargetParameter
        get_current_OV;
    case GET_AUDIBLE_COMPOSITION_EFFECT_TAG:
        RtSocketTargetParameter
            get_audible_composition_effect;
    case GET_EFFECTIVE_OV_TAG:
        RtSocketTargetParameter
            get_effective_OV;
    case GET_PERCEPTIBLE_OV_TAG:
        RtSocketTargetParameter
            get_perceptible_OV;
    case GET_STREAM_CHOSEN_TAG:
        RtSocketTargetParameter
            get_stream_chosen;
    case GET_SELECTABILITY_TAG:
        RtSocketTargetParameter
            get_selectability;
    case GET_EFFECTIVE_SELECTABILITY_TAG:
        RtSocketTargetParameter
            get_effective_selectability;
    case GET_SELECTION_STATUS_TAG:
        RtSocketTargetParameter
            get_selection_status;
    case GET_SELECTION_MODE_TAG:
        RtSocketTargetParameter
            get_selection_mode;
    case GET_NUMBER_SELECTED_SOCKETS_TAG:
        RtSocketTargetParameter
            get_number_selected_sockets;
    case GET_SELECTION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG:
        RtSocketTargetParameter
            get_selection_presentation_effect_responsibility;
    case GET_MODIFIABILITY_TAG:
        RtSocketTargetParameter
            get_modifiability;
    case GET_EFFECTIVE_MODIFIABILITY_TAG:
        RtSocketTargetParameter
            get_effective_modifiability;
    case GET_MODIFICATION_STATUS_TAG:
        RtSocketTargetParameter
            get_modification_status;
    case GET_MODIFICATION_MODE_TAG:
        RtSocketTargetParameter
            get_modification_mode;
    case GET_NUMBER_MODIFIED_SOCKETS_TAG:
        RtSocketTargetParameter
            get_number_modified_sockets;
    case GET_MODIFICATION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG:
        RtSocketTargetParameter
            get_modification_presentation_effect_responsibility;
    case GET_CHANNEL_AVAILABILITY_STATUS_TAG:
        ChannelTargetParameter
            get_channel_availability_status;
    case GET_CHANNEL_PERCEPTABILITY_TAG:
        ChannelTargetParameter
            get_channel_perceptability;
};
attribute EvaluatedValue evaluated_value;
};

// Corresponding MHEG datatype: Delay-Targets
//==========================================================================
enum DelayTargetsTag { PERCEPTIBLE_TARGETS_TAG, NULL_CHANNEL_TAG };
union DelayTargets
switch (DelayTargetsTag){
    case PERCEPTIBLE_TARGETS_TAG:
        sequence<RtObjectSocketReference>
            perceptible_targets;
};
```

```
// Corresponding MHEG datatype: Delay-Targets-Macro
//==========================================================================
struct DelayTargetsMacro {
    MacroDefId
        macro_def_id;
    sequence<DelayTargets,1>
        delay_targets;
};

// Corresponding MHEG datatype: Delay-Targets-Parameter
//==========================================================================
enum DelayTargetsParameterTag { DELAY_TARGETS_TAG, DELAY_TARGETS_MACRO_TAG };
union DelayTargetsParameter
switch (DelayTargetsParameterTag){
    case DELAY_TARGETS_TAG:
        DelayTargets
            delay_targets;
    case DELAY_TARGETS_MACRO_TAG:
        DelayTargetsMacro
            delay_targets_macro;
};

// Corresponding MHEG datatype: Delay
//==========================================================================
struct Delay {
    sequence<DelayTargetsParameter,1>
        targets_parameter;
    NumericParameter
        duration;
};

// Corresponding MHEG datatype: Values
//==========================================================================
typedef sequence<GenericValue> Values;

// Corresponding MHEG datatype: Values-Macro
//==========================================================================
struct ValuesMacro {
    MacroDefId
        macro_def_if;
    sequence<Values,1>
        values;
};

// Corresponding MHEG datatype: Returned-Values-Parameter
//==========================================================================
enum ReturnedValuesParameterTag { VALUES_TAG, VALUES_MACRO_TAG };
union ReturnedValuesParameter
switch (ReturnedValuesParameterTag){
    case VALUES_TAG:
        Values
            values;
    case VALUES_MACRO_TAG:
        ValuesMacro
            values_macro;
};

// Corresponding MHEG datatype: Return
//==========================================================================
struct Return {
    NumericParameter
        return_indicator_param;
    sequence<ReturnedValuesParameter,1>
        returned_values_param;
    sequence<MhTargetsParameter,1>
        returned_objects_param;
};

// Corresponding MHEG datatype: Set-Alias
//==========================================================================
struct SetAlias {
    sequence<TargetsParameter,1>
        targets_parameter;
    StringParameter
        given_alias;
};

// Corresponding MHEG datatype: Data-Element
//===========================================================================
struct DataElement {
    sequence<ElementListIndexParameter,1>
        element_list_index_param;
    sequence<BooleanParameter,1>
        process_indicator_param;
    sequence<ValueParameter,1>
        value_parameter;
};

// Corresponding MHEG datatype: Set-Data
//===========================================================================
struct SetData {
      sequence<MhTargetsParameter,1>
         content_targets_param;
      sequence<BooleanParameter,1>
         substitution_indicator_param;
      sequence<sequence<DataElement>,1>
         data_elements;
};

// Corresponding MHEG datatype: Copy
//===========================================================================
struct Copy {
    sequence<MhTargetParameter,1>
        source;
    MhTargetsParameter
        copies;
};

// Corresponding MHEG datatype: Stream
//===========================================================================
struct Stream {
    sequence<long>
        stream_id;
    MhTargetParameter
        content_target;
};

// Corresponding MHEG datatype: Set-Multiplex
//===========================================================================
struct SetMultiplex {
    sequence<MhTargetsParameter,1>
        multiplex_targets;
    sequence<Stream>
        streams;
};

// Corresponding MHEG datatype: Set-Demultiplex
//===========================================================================
struct SetDemultiplex {
    sequence<MhTargetParameter,1>
        multiplex_targets;
    sequence<Stream>
        multiplex;
};

// Corresponding MHEG datatype: Global-Behaviour
//===========================================================================
enum GlobalBehaviourTag { GLOBAL_BEHAVIOUR_GET_GLOBAL_BEHAVIOUR_TAG,
GLOBAL_BEHAVIOUR_GET_DATA_TAG, GLOBAL_BEHAVIOUR_UNSPECIFIED_TAG };
union GlobalBehaviour
switch (GlobalBehaviourTag){
    case GLOBAL_BEHAVIOUR_GET_GLOBAL_BEHAVIOUR_TAG:
        RtTargetParameter
            get_global_behaviour;
    case GLOBAL_BEHAVIOUR_GET_DATA_TAG:
        GetData
            get_data;
};

// Corresponding MHEG datatype: Global-Behaviour-Macro
//==========================================================================
struct GlobalBehaviourMacro {
    MacroDefId
        macro_def_id;
    sequence<GlobalBehaviour,1>
        global_behaviour;
};

// Corresponding MHEG datatype: Global-Behaviour-Parameter
//==========================================================================
enum GlobalBehaviourParameterTag { GLOBAL_BEHAVIOUR_TAG,
GLOBAL_BEHAVIOUR_MACRO_TAG };
union GlobalBehaviourParameter
switch (GlobalBehaviourParameterTag){
    case GLOBAL_BEHAVIOUR_TAG:
        GlobalBehaviour
            global_behaviour;
    case GLOBAL_BEHAVIOUR_MACRO_TAG:
        GlobalBehaviourMacro
            global_behaviour_macro;
};

// Corresponding MHEG datatype: Set-Global-Behaviour
//==========================================================================
struct SetGlobalBehaviour {
    sequence<RtSocketTargetsParameter,1>
        rt_sockets_targets_param;
    sequence<GlobalBehaviourParameter,1>
        global_behaviour_param;
};

// Corresponding MHEG datatype: Parameter
//==========================================================================
enum ParameterTag { PARAMETER_GENERIC_VALUE_TAG, PARAMETER_CONTENT_TARGET_TAG };
union Parameter
switch (ParameterTag){
    case PARAMETER_GENERIC_VALUE_TAG:
        ValueParameter
            generic_value;
    case PARAMETER_CONTENT_TARGET_TAG:
        MhTargetParameter
            content_target;
};

// Corresponding MHEG datatype: Parameters-Macro
//==========================================================================
struct ParametersMacro {
    MacroDefId
        macro_def_id;
    sequence<sequence<Parameter>,1>
        parameters;
};

// Corresponding MHEG datatype: Parameters-Parameter
//==========================================================================
enum ParametersParameterTag { PARAMETERS_TAG, PARAMETERS_MACRO_TAG };
union ParametersParameter
switch (ParametersParameterTag){
    case PARAMETERS_TAG:
        sequence<Parameter>
            parameters;
    case PARAMETERS_MACRO_TAG:
        ParametersMacro
            parameters_macro;
};
```

```
// Corresponding MHEG datatype: Set-Parameters
//==========================================================================
struct SetParameters {
    sequence<RtTargetsParameter,1>
        rt_script_targets_parameter;
    ParametersParameter
        parameters;
};

// Corresponding MHEG datatype: Plug-In
//==========================================================================
enum PlugInTag { RT_COMPONENT_REFERENCE_TAG, COMPONENT_REFERENCE_TAG, LABEL_TAG
};
union PlugIn
switch (PlugInTag){
    case RT_COMPONENT_REFERENCE_TAG:
        RtObjectReference
            rt_component_reference;
    case COMPONENT_REFERENCE_TAG:
        MhObjectReference
            component_reference;
    case LABEL_TAG:
        GenericString
            label;
};

// Corresponding MHEG datatype: Plug-In-Macro
//==========================================================================
struct PlugInMacro {
    MacroDefId
        macro_def_id;
    sequence<PlugIn,1>
        plug_in;
};

// Corresponding MHEG datatype: Plug-In-Parameter
//==========================================================================
enum PlugInParameterTag { PLUG_IN_TAG, PLUG_IN_MACRO_TAG };
union PlugInParameter
switch (PlugInParameterTag){
    case PLUG_IN_TAG:
        PlugIn
            plug_in;
    case PLUG_IN_MACRO_TAG:
        PlugInMacro
            plug_in_macro;
};

// Corresponding MHEG datatype: Plug
//==========================================================================
struct Plug {
    sequence<SocketTargetsParameter,1>
        socket_targets_parameter;
    PlugInParameter
        plug_in_parameter;
};

// Corresponding MHEG datatype: Set-RGS
//==========================================================================
struct SetRGS {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_param;
    sequence<ChannelTargetParameter,1>
        rgs_parameter;
};

// Corresponding MHEG datatype: Set-Opacity
```

```
//==========================================================================
struct SetOpacity {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_param;
    sequence<NumericParameter,1>
        opacity_rate;
    sequence<NumericParameter,1>
        transition_duration_param;
};

// Corresponding MHEG datatype: Presentation-Priority
//==========================================================================
enum PresentationPriorityTag { PRESENTATION_PRIORITY_GENERIC_NUMERIC_TAG,
PRESENTATION_PRIORITY_PRIORITY_LEVEL_TAG };
union PresentationPriority
switch (PresentationPriorityTag){
    case PRESENTATION_PRIORITY_GENERIC_NUMERIC_TAG:
        GenericNumeric
            generic_numeric;
    case PRESENTATION_PRIORITY_PRIORITY_LEVEL_TAG:
        PriorityLevel
            priority_level;
};

// Corresponding MHEG datatype: Presentation-Priority-Macro
//==========================================================================
struct PresentationPriorityMacro {
    MacroDefId
        macro_def_id;
    sequence<PresentationPriority,1>
        presentation_priority;
};

// Corresponding MHEG datatype: Presentation-Priority-Parameter
//==========================================================================
enum PresentationPriorityParameterTag { PRESENTATION_PRIORITY_TAG,
PRESENTATION_PRIORITY_MACRO_TAG };
union PresentationPriorityParameter
switch (PresentationPriorityParameterTag){
    case PRESENTATION_PRIORITY_TAG:
        PresentationPriority
            presentation_priority;
    case PRESENTATION_PRIORITY_MACRO_TAG:
        PresentationPriorityMacro
            presentation_priority_macro;
};

// Corresponding MHEG datatype: Set-Presentation-Priority
//==========================================================================
struct SetPresentationPriority {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_param;
    sequence<PresentationPriorityParameter,1>
        presentation_priority_param;
    sequence<NumericParameter,1>
        transition_duration_param;
};

// Corresponding MHEG datatype: Temporal-Position-Macro
//==========================================================================
struct TemporalPositionMacro {
    MacroDefId
        macro_def_id;
    sequence<TemporalPosition,1>
        temporal_position;
};

// Corresponding MHEG datatype: Temporal-Position-Parameter
//==========================================================================
enum TemporalPositionParameterTag { TEMPORAL_POSITION_TAG,
TEMPORAL_POSITION_MACRO_TAG };
union TemporalPositionParameter
```

```
switch (TemporalPositionParameterTag){
    case TEMPORAL_POSITION_TAG:
        TemporalPosition
            temporal_position;
    case TEMPORAL_POSITION_MACRO_TAG:
        TemporalPositionMacro
            temporal_position_macro;
};

// Corresponding MHEG datatype: Set-Visible-Duration
//=========================================================================
struct SetVisibleDuration {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<TemporalPositionParameter,1>
        initial_temporal_position_parameter;
    sequence<TemporalPositionParameter,1>
        terminal_temporal_position_parameter;
};

// Corresponding MHEG datatype: Temporal-Termination-Macro
//=========================================================================
struct TemporalTerminationMacro {
    MacroDefId
        macro_def_id;
    sequence<TemporalTermination,1>
        temporal_termination;
};

// Corresponding MHEG datatype: Temporal-Termination-Parameter
//=========================================================================
enum TemporalTerminationParameterTag {
TEMPORAL_TERMINATION_PARAMETER_TEMPORAL_TERMINATION_TAG,
TEMPORAL_TERMINATION_PARAMETER_TEMPORAL_TERMINATION_MACRO_TAG };
union TemporalTerminationParameter
switch (TemporalTerminationParameterTag){
    case TEMPORAL_TERMINATION_PARAMETER_TEMPORAL_TERMINATION_TAG:
        TemporalTermination
            temporal_termination;
    case TEMPORAL_TERMINATION_PARAMETER_TEMPORAL_TERMINATION_MACRO_TAG:
        TemporalTerminationMacro
            temporal_termination_macro;
};

// Corresponding MHEG datatype: Set-Temporal-Termination
//=========================================================================
struct SetTemporalTermination {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    TemporalTerminationParameter
        temporal_termination_parameter;
};

// Corresponding MHEG datatype: Current-Temporal-Position-Macro
//=========================================================================
struct CurrentTemporalPositionMacro {
    MacroDefId
        macro_def_id;
    sequence<CurrentTemporalPosition,1>
        current_temporal_termination;
};

// Corresponding MHEG datatype: Current-Temporal-Position-Parameter
//=========================================================================
enum CurrentTemporalPositionParameterTag { CURRENT_TEMPORAL_POSITION_TAG,
CURRENT_TEMPORAL_POSITION_MACRO_TAG };
union CurrentTemporalPositionParameter
switch (CurrentTemporalPositionParameterTag){
    case CURRENT_TEMPORAL_POSITION_TAG:
        CurrentTemporalPosition
            current_temporal_position;
    case CURRENT_TEMPORAL_POSITION_MACRO_TAG:
        CurrentTemporalPositionMacro
            current_temporal_position_macro;
};
```

```
// Corresponding MHEG datatype: Set-Current-Temporal-Position
//==========================================================================
struct SetCurrentTemporalPosition {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<CurrentTemporalPositionParameter,1>
        temporal_position_parameter;
};

// Corresponding MHEG datatype: Visible-Duration-Macro
//==========================================================================
struct VisibleDurationPositionMacro {
    MacroDefId
        macro_def_id;
    sequence<VisibleDurationPosition,1>
        visible_duration_position;
};

// Corresponding MHEG datatype: Visible-Duration-Parameter
//==========================================================================
enum VisibleDurationPositionParameterTag { VISIBLE_DURATION_POSITION_TAG,
VISIBLE_DURATION_POSITION_MACRO_TAG };
union VisibleDurationPositionParameter
switch (VisibleDurationPositionParameterTag){
    case VISIBLE_DURATION_POSITION_TAG:
        VisibleDurationPosition
            visible_duration_position;
    case VISIBLE_DURATION_POSITION_MACRO_TAG:
        VisibleDurationPositionMacro
            visible_duration_position_macro;
};

// Corresponding MHEG datatype: Set-Visible-Duration-Position
//==========================================================================
struct SetVisibleDurationPosition {
    sequence<SocketTargetsParameter,1>
        socket_targets;
    sequence<VisibleDurationPositionParameter,1>
        visible_duration_position_param;
};

// Corresponding MHEG datatype: Speed-Macro
//==========================================================================
struct SpeedMacro {
    MacroDefId
        macro_def_id;
    sequence<Speed,1>
        speed;
};

// Corresponding MHEG datatype: Speed-Parameter
//==========================================================================
enum SpeedParameterTag { SPEED_TAG, SPEED_MACRO_TAG };
union SpeedParameter
switch (SpeedParameterTag){
    case SPEED_TAG:
        Speed
            speed;
    case SPEED_MACRO_TAG:
        SpeedMacro
            speed_macro;
};

// Corresponding MHEG datatype: Set-Speed
//==========================================================================
struct SetSpeed {
    RtSocketTargetsParameter
        perceptible_targets_parameter;
    sequence<SpeedParameter,1>
        speed_parameter;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};
```

```
// Corresponding MHEG datatype: Timestone-Macro
//==========================================================================
struct TimestoneMacro {
    MacroDefId
        macro_def_id;
    sequence<Timestone,1>
        timestone;
};

// Corresponding MHEG datatype: Timestone-Parameter
//==========================================================================
enum TimestoneParameterTag { TIMESTONE_TAG, TIMESTONE_MACRO_TAG };
union TimestoneParameter
switch (TimestoneParameterTag){
    case TIMESTONE_TAG:
        Timestone
            timestone;
    case TIMESTONE_MACRO_TAG:
        TimestoneMacro
            timestone_macro;
};

// Corresponding MHEG datatype: Set-Timestones
//==========================================================================
struct SetTimestones {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<sequence<TimestoneParameter>,1>
        timestones_parameter;
};

// Corresponding MHEG datatype: Perceptible-Size-Projection-Macro
//==========================================================================
struct PerceptibleSizeProjectionMacro {
    MacroDefId
        macro_def_id;
    sequence<PerceptibleSizeProjection,1>
        perceptible_size_projection;
};

// Corresponding MHEG datatype: Perceptible-Size-Projection-Parameter
//==========================================================================
enum PerceptibleSizeProjectionParameterTag { PERCEPTIBLE_SIZE_PROJECTION_TAG,
PERCEPTIBLE_SIZE_PROJECTION_MACRO_TAG };
union PerceptibleSizeProjectionParameter
switch (PerceptibleSizeProjectionParameterTag){
    case PERCEPTIBLE_SIZE_PROJECTION_TAG:
        PerceptibleSizeProjection
            perceptible_size_projection;
    case PERCEPTIBLE_SIZE_PROJECTION_MACRO_TAG:
        PerceptibleSizeProjectionMacro
            perceptible_size_projection_macro;
};

// Corresponding MHEG datatype: Set-Perceptible-Size-Projection
//==========================================================================
struct SetPerceptibleSizeProjection {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    PerceptibleSizeProjectionParameter
        perceptible_size_projection;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};

// Corresponding MHEG datatype: Resizing-Strategy-Macro
//==========================================================================
struct ResizingStrategyMacro {
    MacroDefId
        macro_def_id;
    sequence<ResizingStrategy,1>
        resizing_strategy;
};
```

```
// Corresponding MHEG datatype: Resizing-Strategy-Parameter
//==========================================================================
enum ResizingStrategyParameterTag {
RESIZING_STRATEGY_PARAMETER_RESIZING_STRATEGY_TAG,
RESIZING_STRATEGY_PARAMETER_RESIZING_STRATEGY_MACRO_TAG };
union ResizingStrategyParameter
switch (ResizingStrategyParameterTag){
    case RESIZING_STRATEGY_PARAMETER_RESIZING_STRATEGY_TAG:
        ResizingStrategy
            resizing_strategy;
    case RESIZING_STRATEGY_PARAMETER_RESIZING_STRATEGY_MACRO_TAG:
        ResizingStrategyMacro
            resizing_strategy_macro;
};

// Corresponding MHEG datatype: Set-Resizing-Strategy
//==========================================================================
struct SetResizingStrategy {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<ResizingStrategyParameter,1>
        resizing_strategy_parameter;
};

// Corresponding MHEG datatype: Set-Aspect-Ratio-Preserved
//==========================================================================
struct SetAspectRatioPreserved {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    BooleanParameter
        preserved;
};

// Corresponding MHEG datatype: VSGS-Macro
//==========================================================================
struct VSGSMacro {
    MacroDefId
        macro_def_id;
    sequence<VSGS,1>
        vsgs;
};

// Corresponding MHEG datatype: VSGS-Parameter
//==========================================================================
enum VSGSParameterTag { VSGS_PARAMETER_VSGS_TAG, VSGS_PARAMETER_VSGS_MACRO_TAG };
union VSGSParameter
switch (VSGSParameterTag){
    case VSGS_PARAMETER_VSGS_TAG:
        VSGS
            vsgs;
    case VSGS_PARAMETER_VSGS_MACRO_TAG:
        VSGSMacro
            vsgs_macro;
};

// Corresponding MHEG datatype: Size-Attribute-Macro
//==========================================================================
struct SizeAttributeMacro {
    MacroDefId
        macro_def_id;
    sequence<SizeAttribute,1>
        size_attribute;
};

// Corresponding MHEG datatype: Size-Attribute-Parameter
//==========================================================================
enum SizeAttributeParameterTag { SIZE_ATTRIBUTE_TAG, SIZE_ATTRIBUTE_MACRO_TAG };
union SizeAttributeParameter
switch (SizeAttributeParameterTag){
    case SIZE_ATTRIBUTE_TAG:
        SizeAttribute
            size_attribute;
    case SIZE_ATTRIBUTE_MACRO_TAG:
        SizeAttributeMacro
            size_attribute_macro;
};
// Corresponding MHEG datatype: VS-Parameter
//=========================================================================
enum VSParameterTag { X_SIZE_ATTRIBUTE_PARAMETER_TAG,
Y_SIZE_ATTRIBUTE_PARAMETER_TAG, Z_SIZE_ATTRIBUTE_PARAMETER_TAG };
union VSParameter
switch (VSParameterTag){
    case X_SIZE_ATTRIBUTE_PARAMETER_TAG:
        SizeAttributeParameter
            x_size_attribute_parameter;
    case Y_SIZE_ATTRIBUTE_PARAMETER_TAG:
        SizeAttributeParameter
            y_size_attribute_parameter;
    case Z_SIZE_ATTRIBUTE_PARAMETER_TAG:
        SizeAttributeParameter
            z_size_attribute_parameter;
};
// Corresponding MHEG datatype: Set-Visible-Size
//=========================================================================
struct SetVisibleSize {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    VSGSParameter
        vsgs_parameter;
    VSParameter
        vs_parameter;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};
// Corresponding MHEG datatype: Adjustment-Axis-Macro
//=========================================================================
struct AdjustmentAxisMacro {
    MacroDefId
        macro_def_id;
    sequence<AdjustmentAxis,1>
        adjustment_axis;
};
// Corresponding MHEG datatype: Adjustment-Axis-Parameter
//=========================================================================
enum AdjustmentAxisParameterTag { ADJUSTMENT_AXIS_TAG, ADJUSTMENT_AXIS_MACRO_TAG
};
union AdjustmentAxisParameter
switch (AdjustmentAxisParameterTag){
    case ADJUSTMENT_AXIS_TAG:
        AdjustmentAxis
            adjustment_axis;
    case ADJUSTMENT_AXIS_MACRO_TAG:
        AdjustmentAxisMacro
            adjustment_axis_macro;
};
// Corresponding MHEG datatype: Adjustment-Policy-Macro
//=========================================================================
struct AdjustmentPolicyMacro {
    MacroDefId
        macro_def_id;
    sequence<AdjustmentPolicy,1>
        adjustment_policy;
};
// Corresponding MHEG datatype: Adjustment-Policy-Parameter
//=========================================================================
enum AdjustmentPolicyParameterTag { ADJUSTMENT_POLICY_TAG,
ADJUSTMENT_POLICY_MACRO_TAG };
union AdjustmentPolicyParameter
switch (AdjustmentPolicyParameterTag){
    case ADJUSTMENT_POLICY_TAG:
        AdjustmentPolicy
            adjustment_policy;
    case ADJUSTMENT_POLICY_MACRO_TAG:
        AdjustmentPolicyMacro
            adjustment_policy_macro;
};

// Corresponding MHEG datatype: Set-Visible-Sizes-Adjustment
//==========================================================================
struct SetVisibleSizesAdjustment {
    sequence<AdjustmentAxisParameter>
        adjustment_axis_set;
    AdjustmentPolicyParameter
        adjustment_policy;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};

// Corresponding MHEG datatype: Set-Box
//==========================================================================
struct SetBox {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<BooleanParameter,1>
        box_parameter;
};

// Corresponding MHEG datatype: Set-Default-Background
//==========================================================================
struct SetDefaultBackground {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<NumericParameter,1>
        background_parameter;
    sequence<NumericParameter,1>
        transition_duration;
};

// Corresponding MHEG datatype: Attachment-Point-Type-Macro
//==========================================================================
struct AttachmentPointTypeMacro {
    MacroDefId
        macro_def_id;
    sequence<AttachmentPointType,1>
        attachment_point_type;
};

// Corresponding MHEG datatype: Attachment-Point-Type-Parameter
//==========================================================================
enum AttachmentPointTypeParameterTag { ATTACHMENT_POINT_TYPE_TAG,
ATTACHMENT_POINT_TYPE_MACRO_TAG };
union AttachmentPointTypeParameter
switch (AttachmentPointTypeParameterTag){
    case ATTACHMENT_POINT_TYPE_TAG:
        AttachmentPointType
            attachment_point_type;
    case ATTACHMENT_POINT_TYPE_MACRO_TAG:
        AttachmentPointTypeMacro
            attachment_point_type_macro;
};

// Corresponding MHEG datatype: Attachment-Attribute-Macro
//==========================================================================
struct AttachmentAttributeMacro {
    MacroDefId
        macro_def_id;
    sequence<AttachmentAttribute,1>
        attachment_attribute;
};
```

```
// Corresponding MHEG datatype: Attachment-Attribute-Parameter
//==========================================================================
enum AttachmentAttributeParameterTag { ATTACHMENT_ATTRIBUTE_TAG,
ATTACHMENT_ATTRIBUTE_MACRO_TAG };
union AttachmentAttributeParameter
switch (AttachmentAttributeParameterTag){
    case ATTACHMENT_ATTRIBUTE_TAG:
        AttachmentAttribute
            attachment_attribute;
    case ATTACHMENT_ATTRIBUTE_MACRO_TAG:
        AttachmentAttributeMacro
            attachment_attribute_macro;
};

// Corresponding MHEG datatype: Attachment-Point-Parameter
//==========================================================================
struct AttachmentPointParameter {
    sequence<AttachmentAttributeParameter,1>
        x_attachment_parameter;
    sequence<AttachmentAttributeParameter,1>
        y_attachment_parameter;
    sequence<AttachmentAttributeParameter,1>
        z_attachment_parameter;
};

// Corresponding MHEG datatype: Set-Attachment-Point
//==========================================================================
struct SetAttachmentPoint {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<AttachmentPointTypeParameter,1>
        attachment_point_type_param;
    sequence<AttachmentPointParameter,1>
        attachment_point_param;
};

// Corresponding MHEG datatype: Reference-Type-Macro
//==========================================================================
struct ReferenceTypeMacro {
    MacroDefId
        macro_def_id;
    sequence<ReferenceType,1>
        reference_type;
};

// Corresponding MHEG datatype: Reference-Type-Parameter
//==========================================================================
enum ReferenceTypeParameterTag { REFERENCE_TYPE_TAG, REFERENCE_TYPE_MACRO_TAG };
union ReferenceTypeParameter
switch (ReferenceTypeParameterTag){
    case REFERENCE_TYPE_TAG:
        ReferenceType
            reference_type;
    case REFERENCE_TYPE_MACRO_TAG:
        ReferenceTypeMacro
            reference_type_macro;
};

// Corresponding MHEG datatype: VSEAP-Reference-Point
//==========================================================================
enum VSEAPReferencePointTag { ORIGIN_TAG, COMPONENT_TAG };
union VSEAPReferencePoint
switch (VSEAPReferencePointTag){
    case COMPONENT_TAG:
        SocketTargetParameter
            component;
};

// Corresponding MHEG datatype: VSEAP-Reference-Point-Macro
//==========================================================================
struct VSEAPReferencePointMacro {
    MacroDefId
        macro_def_id;
    sequence<VSEAPReferencePoint,1>
        vseap_reference;
};

// Corresponding MHEG datatype: VSEAP-Reference-Parameter
//======================================================================
enum VSEAPReferenceParameterTag { VSEAP_REFERENCE_TAG, VSEAP_REFERENCE_MACRO_TAG
};
union VSEAPReferenceParameter
switch (VSEAPReferenceParameterTag){
    case VSEAP_REFERENCE_TAG:
        VSEAPReferencePoint
            vseap_reference;
    case VSEAP_REFERENCE_MACRO_TAG:
        VSEAPReferencePointMacro
            vseap_reference_macro;
};

// Corresponding MHEG datatype: Length-Attribute-Macro
//======================================================================
struct LengthAttributeMacro {
    MacroDefId
        macro_def_id;
    sequence<LengthAttribute,1>
        length_attribute;
};

// Corresponding MHEG datatype: Length-Attribute-Parameter
//======================================================================
enum LengthAttributeParameterTag { LENGTH_ATTRIBUTE_TAG,
LENGTH_ATTRIBUTE_MACRO_TAG };
union LengthAttributeParameter
switch (LengthAttributeParameterTag){
    case LENGTH_ATTRIBUTE_TAG:
        LengthAttribute
            length_attribute;
    case LENGTH_ATTRIBUTE_MACRO_TAG:
        LengthAttributeMacro
            length_attribute_macro;
};

// Corresponding MHEG datatype: Lengths-Parameter
//======================================================================
struct LengthsParameter {
    sequence<LengthAttributeParameter,1>
        x_length_parameter;
    sequence<LengthAttributeParameter,1>
        y_position_parameter;
    sequence<LengthAttributeParameter,1>
        z_position_parameter;
};

// Corresponding MHEG datatype: Set-Attachment-Point-Position
//======================================================================
struct SetAttachmentPointPosition {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<ReferenceTypeParameter,1>
        attachment_type_parameter;
    sequence<VSEAPReferenceParameter,1>
        vseap_reference_parameter;
    sequence<LengthsParameter,1>
        positions_parameter;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};

// Corresponding MHEG datatype: Size-Border-Macro
//======================================================================
struct SizeBorderMacro {
    MacroDefId
        macro_def_id;
    sequence<SizeBorder,1>
        size_border;
};
```

```
// Corresponding MHEG datatype: Size-Border-Parameter
//==========================================================================
enum SizeBorderParameterTag { SIZE_BORDER_TAG, SIZE_BORDER_MACRO_TAG };
union SizeBorderParameter
switch (SizeBorderParameterTag){
    case SIZE_BORDER_TAG:
        SizeBorder
            size_border;
    case SIZE_BORDER_MACRO_TAG:
        SizeBorderMacro
            size_border_macro;
};

// Corresponding MHEG datatype: Set-Visible-Sizes-Alignment
//==========================================================================
struct SetVisibleSizesAlignment {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<SizeBorderParameter,1>
        size_border_parameter;
    sequence<NumericParameter,1>
        margin_parameter;
    sequence<NumericParameter,1>
        transition_duration;
};

// Corresponding MHEG datatype: User-Controls-Macro
//==========================================================================
struct UserControlsMacro {
    MacroDefId
        macro_def_id;
    sequence<UserControls,1>
        user_controls;
};

// Corresponding MHEG datatype: User-Controls-Parameter
//==========================================================================
enum UserControlsParameterTag { USER_CONTROLS_PARAMETER_USER_CONTROLS_TAG,
USER_CONTROLS_PARAMETER_USER_CONTROLS_MACRO_TAG };
union UserControlsParameter
switch (UserControlsParameterTag){
    case USER_CONTROLS_PARAMETER_USER_CONTROLS_TAG:
        UserControls
            user_controls;
    case USER_CONTROLS_PARAMETER_USER_CONTROLS_MACRO_TAG:
        UserControlsMacro
            user_controls_macro;
};

// Corresponding MHEG datatype: Set-Moving-Ability
//==========================================================================
struct SetMovingAbility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<UserControlsParameter,1>
        allowed;
};

// Corresponding MHEG datatype: Set-Resizing-Ability
//==========================================================================
struct SetResizingAbility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<UserControlsParameter,1>
        allowed;
};

// Corresponding MHEG datatype: Set-Scaling-Ability
//==========================================================================
struct SetScalingAbility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<UserControlsParameter,1>
        allowed;
};
```

```
// Corresponding MHEG datatype: Set-Scrolling-Ability
//==========================================================================
struct SetScrollingAbility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<UserControlsParameter,1>
        allowed;
};

// Corresponding MHEG datatype: Audible-Volume-Macro
//==========================================================================
struct AudibleVolumeMacro {
    MacroDefId
        macro_def_id;
    sequence<AudibleVolume,1>
        audible_volume;
};

// Corresponding MHEG datatype: Audible-Volume-Parameter
//==========================================================================
enum AudibleVolumeParameterTag { AUDIBLE_VOLUME_TAG, AUDIBLE_VOLUME_MACRO_TAG };
union AudibleVolumeParameter
switch (AudibleVolumeParameterTag){
    case AUDIBLE_VOLUME_TAG:
        AudibleVolume
            audible_volume;
    case AUDIBLE_VOLUME_MACRO_TAG:
        AudibleVolumeMacro
            audible_volume_macro;
};

// Corresponding MHEG datatype: Set-Audible-Volume
//==========================================================================
struct SetAudibleVolume {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<AudibleVolumeParameter,1>
        audible_volume_parameter;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};

// Corresponding MHEG datatype: Set-Audible-Composition-Effect
//==========================================================================
struct SetAudibleCompositionEffect {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<NumericParameter,1>
        audible_effect_parameter;
    sequence<NumericParameter,1>
        transition_duration_parameter;
};

// Corresponding MHEG datatype: Stream-Identifier-Macro
//==========================================================================
struct StreamIdentifierMacro {
    MacroDefId
        macro_def_id;
    sequence<StreamIdentifier,1>
        stream_identifier;
};

// Corresponding MHEG datatype: Stream-Identifier-Parameter
//==========================================================================
enum StreamIdentifierParameterTag { STREAM_IDENTIFIER_TAG,
STREAM_IDENTIFIER_MACRO_TAG };
union StreamIdentifierParameter
switch (StreamIdentifierParameterTag){
    case STREAM_IDENTIFIER_TAG:
        StreamIdentifier
            stream_identifier;
    case STREAM_IDENTIFIER_MACRO_TAG:
        StreamIdentifierMacro
            stream_identifier_macro;
};
```

```
// Corresponding MHEG datatype: Set-Stream-Choice
//==========================================================================
struct SetStreamChoice {
    RtSocketTargetsParameter
        rt_multiplexed_content_socket_param;
    sequence<StreamIdentifierParameter,1>
        stream_identifier_parameter;
};

// Corresponding MHEG datatype: Set-Selectability
//==========================================================================
struct SetSelectability {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<NumericParameter,1>
        min_number_selections;
    sequence<NumericParameter,1>
        max_number_selections;
};

// Corresponding MHEG datatype: Selection-State-Macro
//==========================================================================
struct SelectionStateMacro {
    MacroDefId
        macro_def_id;
    sequence<SelectionStatusValue,1>
        modification_state;
};

// Corresponding MHEG datatype: Selection-State-Parameter
//==========================================================================
enum SelectionStateParameterTag { SELECTION_STATE_MACRO_TAG, SELECTION_STATE_TAG
};
union SelectionStateParameter
switch (SelectionStateParameterTag){
    case SELECTION_STATE_MACRO_TAG:
        SelectionStateMacro
            selection_state_macro;
    case SELECTION_STATE_TAG:
        SelectionStatusValue
            selection_state;
};

// Corresponding MHEG datatype: Set-Selection-Status
//==========================================================================
struct SetSelectionStatus {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<SelectionStateParameter,1>
        selection_state_parameter;
};

// Corresponding MHEG datatype: Responsibility-Macro
//==========================================================================
struct ResponsibilityMacro {
    MacroDefId
        macro_def_id;
    sequence<Responsibility,1>
        responsibility;
};

// Corresponding MHEG datatype: Responsibility-Parameter
//==========================================================================
enum ResponsibilityParameterTag { RESPONSIBILITY_PARAMETER_RESPONSIBILITY_TAG,
RESPONSIBILITY_PARAMETER_RESPONSIBILITY_MACRO_TAG };
union ResponsibilityParameter
switch (ResponsibilityParameterTag){
    case RESPONSIBILITY_PARAMETER_RESPONSIBILITY_TAG:
        Responsibility
            responsibility;
    case RESPONSIBILITY_PARAMETER_RESPONSIBILITY_MACRO_TAG:
        ResponsibilityMacro
            responsibility_macro;
};
```

```
// Corresponding MHEG datatype: Set-Selection-Presentation-Effect-Responsibility
//=========================================================================
struct SetSelectionPresentationEffectResponsibility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<ResponsibilityParameter,1>
        selection_responsibility;
};

// Corresponding MHEG datatype: Set-Modifiability
//=========================================================================
struct SetModifiability {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<NumericParameter,1>
        min_number_modifications;
    sequence<NumericParameter,1>
        max_number_modifications;
};

// Corresponding MHEG datatype: Modification-State-Macro
//=========================================================================
struct ModificationStateMacro {
    MacroDefId
        macro_def_id;
    sequence<ModificationStatusValue,1>
        modification_state;
};

// Corresponding MHEG datatype: Modification-State-Parameter
//=========================================================================
enum ModificationStateParameterTag { MODIFICATION_STATE_MACRO_TAG,
MODIFICATION_STATE_TAG };
union ModificationStateParameter
switch (ModificationStateParameterTag){
    case MODIFICATION_STATE_MACRO_TAG:
        ModificationStateMacro
            modification_state_macro;
    case MODIFICATION_STATE_TAG:
        ModificationStatusValue
            modification_state;
};

// Corresponding MHEG datatype: Set-Modification-Status
//=========================================================================
struct SetModificationStatus {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<ModificationStateParameter,1>
        modification_state_parameter;
};

// Corresponding MHEG datatype: Set-Modification-Presentation-Effect-
//   Responsibility
//=========================================================================
struct SetModificationPresentationEffectResponsibility {
    sequence<RtSocketTargetsParameter,1>
        perceptible_targets_parameter;
    sequence<ResponsibilityParameter,1>
        modification_responsibility;
};

// Corresponding MHEG datatype: Presentation-State-Macro
//=========================================================================
struct PresentationStateMacro {
    MacroDefId
        macro_def_id;
    sequence<PresentationState,1>
        presentation_state;
};
```

```
// Corresponding MHEG datatype: Presentation-State-Parameter
//=========================================================================
enum PresentationStateParameterTag { PRESENTATION_STATE_MACRO_TAG,
PRESENTATION_STATE_TAG };
union PresentationStateParameter
switch (PresentationStateParameterTag){
    case PRESENTATION_STATE_MACRO_TAG:
        PresentationStateMacro
            presentation_state_macro;
    case PRESENTATION_STATE_TAG:
        PresentationState
            presentation_state;
};

// Corresponding MHEG datatype: Alternate-Presentation-State-Macro
//=========================================================================
struct AlternatePresentationMacro {
    MacroDefId
        macro_def_id;
    sequence<AlternatePresentation,1>
        alternate_presentation_state;
};

// Corresponding MHEG datatype: Alternate-Presentation-State-Parameter
//=========================================================================
enum AlternatePresentationParameterTag { ALTERNATE_PRESENTATION_MACRO_TAG,
ALTERNATE_PRESENTATION_TAG };
union AlternatePresentationParameter
switch (AlternatePresentationParameterTag){
    case ALTERNATE_PRESENTATION_MACRO_TAG:
        AlternatePresentationMacro
            alternate_presentation_macro;
    case ALTERNATE_PRESENTATION_TAG:
        AlternatePresentation
            alternate_presentation;
};

// Corresponding MHEG datatype: Set-Button-Style
//=========================================================================
struct SetButtonStyle {
    sequence<RtSocketTargetsParameter,1>
        targets;
    sequence<PresentationStateParameter,1>
        initial_state;
    sequence<AlternatePresentationParameter,1>
        alternate_presentation1;
    sequence<AlternatePresentationParameter,1>
        alternate_presentation2;
    sequence<AlternatePresentationParameter,1>
        alternate_presentation3;
};

// Corresponding MHEG datatype: Orientation-Macro
//=========================================================================
struct OrientationMacro {
    MacroDefId
        macro_def_id;
    sequence<Orientation,1>
        orientation;
};

// Corresponding MHEG datatype: Orientation-Parameter
//=========================================================================
enum OrientationParameterTag { ORIENTATION_TAG, ORIENTATION_MACRO_TAG };
union OrientationParameter
switch (OrientationParameterTag){
    case ORIENTATION_TAG:
        Orientation
            orientation;
    case ORIENTATION_MACRO_TAG:
        OrientationMacro
            orientation_macro;
};
```

```
// Corresponding MHEG datatype: Set-Slider-Style
//==========================================================================
struct SetSliderStyle {
    sequence<RtSocketTargetsParameter,1>
        targets;
    sequence<RtSocketTargetParameter,1>
        cursor;
    sequence<RtSocketTargetParameter,1>
        background;
    sequence<OrientationParameter,1>
        orientation;
    sequence<NumericParameter,1>
        min_value;
    NumericParameter
        max_value;
};

// Corresponding MHEG datatype: Echo-Style-Macro
//==========================================================================
struct EchoStyleMacro {
    MacroDefId
        macro_def_id;
    sequence<EchoStyle,1>
        echo_style;
};

// Corresponding MHEG datatype: Echo-Style-Parameter
//==========================================================================
enum EchoStyleParameterTag { ECHO_STYLE_MACRO_TAG, ECHO_STYLE_TAG };
union EchoStyleParameter
switch (EchoStyleParameterTag){
    case ECHO_STYLE_MACRO_TAG:
        EchoStyleMacro
            echo_style_macro;
    case ECHO_STYLE_TAG:
        EchoStyle
            echo_style;
};

// Corresponding MHEG datatype: Set-Entry-Field-Style
//==========================================================================
struct SetEntryFieldStyle {
    sequence<RtSocketTargetsParameter,1>
        targets;
    sequence<EchoStyleParameter,1>
        echo;
    sequence<RtSocketTargetParameter,1>
        background;
};

// Corresponding MHEG datatype: Association-Macro
//==========================================================================
struct AssociationMacro {
    MacroDefId
        macro_def_id;
    sequence<Association,1>
        association;
};

// Corresponding MHEG datatype: Association-Parameter
//==========================================================================
enum AssociationParameterTag { ASSOCIATION_TAG, ASSOCIATION_MACRO_TAG };
union AssociationParameter
switch (AssociationParameterTag){
    case ASSOCIATION_TAG:
        Association
            association;
    case ASSOCIATION_MACRO_TAG:
        AssociationMacro
            association_macro;
};
```

```
// Corresponding MHEG datatype: Set-Menu-Style
//==========================================================================
struct SetMenuStyle {
    sequence<RtSocketTargetsParameter,1>
        targets;
    sequence<OrientationParameter,1>
        upper_menu_orientation;
    sequence<AssociationParameter>
        associations;
};

// Corresponding MHEG datatype: Socket-Tail-Macro
//==========================================================================
struct SocketTailMacro {
    MacroDefId
        macro_def_id;
    sequence<SocketTail,1>
        socket_tail;
};

// Corresponding MHEG datatype: Socket-Tail-Parameter
//==========================================================================
enum SocketTailParameterTag { SOCKET_TAIL_TAG, SOCKET_TAIL_MACRO_TAG };
union SocketTailParameter
switch (SocketTailParameterTag){
    case SOCKET_TAIL_TAG:
        SocketTail
            socket_tail;
    case SOCKET_TAIL_MACRO_TAG:
        SocketTailMacro
            socket_tail_macro;
};

// Corresponding MHEG datatype: Separator-Macro
//==========================================================================
struct SeparatorMacro {
    MacroDefId
        macro_def_id;
    sequence<Separator,1>
        separator;
};

// Corresponding MHEG datatype: Separator-Parameter
//==========================================================================
enum SeparatorParameterTag { SEPARATOR_MACRO_TAG, SEPARATOR_TAG };
union SeparatorParameter
switch (SeparatorParameterTag){
    case SEPARATOR_MACRO_TAG:
        SeparatorMacro
            separator_macro;
    case SEPARATOR_TAG:
        Separator
            separator;
};

// Corresponding MHEG datatype: Slider-Side-Macro
//==========================================================================
struct SliderSideMacro {
    MacroDefId
        macro_def_id;
    sequence<SliderSide,1>
        slider_side;
};

// Corresponding MHEG datatype: Slider-Side-Parameter
//==========================================================================
enum SliderSideParameterTag { SLIDER_SIDE_TAG, SLIDER_SIDE_MACRO_TAG };
union SliderSideParameter
switch (SliderSideParameterTag){
    case SLIDER_SIDE_TAG:
        SliderSide
            slider_side;
    case SLIDER_SIDE_MACRO_TAG:
        SliderSideMacro
            slider_side_macro;
};

// Corresponding MHEG datatype: Set-Slider-Style-Macro
//==========================================================================
struct SetSliderStyleMacro {
    MacroDefId
        macro_def_id;
    sequence<SetSliderStyle,1>
        set_slider_style;
};

// Corresponding MHEG datatype: Set-Slider-Style-Parameter
//==========================================================================
enum SetSliderStyleParameterTag { SET_SLIDER_STYLE_MACRO_TAG,
SET_SLIDER_STYLE_ACTION_TAG };
union SetSliderStyleParameter
switch (SetSliderStyleParameterTag){
    case SET_SLIDER_STYLE_MACRO_TAG:
        SetSliderStyleMacro
            set_slider_style_macro;
    case SET_SLIDER_STYLE_ACTION_TAG:
        SetSliderStyle
            set_slider_style_action;
};

// Corresponding MHEG datatype: Set-Scrolling-List-Style
//==========================================================================
struct SetScrollingListStyle {
    sequence<RtSocketTargetsParameter,1>
        targets;
    sequence<RtSocketTargetParameter,1>
        background;
    sequence<NumericParameter,1>
        visible_items_number;
    sequence<SocketTailParameter,1>
        first_item;
    sequence<SeparatorParameter,1>
        separator;
    sequence<OrientationParameter,1>
        orientation;
    sequence<SliderSideParameter,1>
        slider_side;
    sequence<SetSliderStyleParameter,1>
        set_slider_style_action;
};

// Corresponding MHEG datatype: Original-Def-Declaration-Macro
//==========================================================================
struct OriginalDefDeclarationMacro {
    MacroDefId
        macro_def_id;
    sequence<OriginalDefDeclaration,1>
        original_definition;
};

// Corresponding MHEG datatype: Original-Def-Declaration-Parameter
//==========================================================================
enum OriginalDefDeclarationParameterTag { ORIGINAL_DEFINITION_TAG,
ORIGINAL_DEFINITION_MACRO_TAG };
union OriginalDefDeclarationParameter
switch (OriginalDefDeclarationParameterTag){
    case ORIGINAL_DEFINITION_TAG:
        OriginalDefDeclaration
            original_definition;
    case ORIGINAL_DEFINITION_MACRO_TAG:
        OriginalDefDeclarationMacro
            original_definition_macro;
};
```

```
// Corresponding MHEG datatype: New-Channel
//==========================================================================
struct NewChannel {
    sequence<ChannelTargetsParameter,1>
        channel_targets_parameter;
    sequence<OriginalDefDeclarationParameter,1>
        original_definition;
};

// Corresponding MHEG datatype: Channel-Perceptability-Values-Macro
//==========================================================================
struct ChannelPerceptabilityValueMacro {
    MacroDefId
        macro_def_id;
    sequence<ChannelPerceptabilityValue,1>
        channel_perceptability_value;
};

// Corresponding MHEG datatype: Perceptability-Parameter
//==========================================================================
enum PerceptabilityParameterTag { CHANNEL_PERCEPTABILITY_VALUE_TAG,
CHANNEL_PERCEPTABILITY_VALUE_MACRO_TAG };
union PerceptabilityParameter
switch (PerceptabilityParameterTag){
    case CHANNEL_PERCEPTABILITY_VALUE_TAG:
        ChannelPerceptabilityValue
            channel_perceptability_value;
    case CHANNEL_PERCEPTABILITY_VALUE_MACRO_TAG:
        ChannelPerceptabilityValueMacro
            channel_perceptability_value_macro;
};

// Corresponding MHEG datatype: Set-Channel-Perceptability
//==========================================================================
struct SetChannelPerceptability {
    sequence<ChannelTargetsParameter,1>
        channel_targets_parameter;
    sequence<PerceptabilityParameter,1>
        perceptability_parameter;
};

// Corresponding MHEG datatype: Elementary-Action
//==========================================================================
enum ElementaryActionTag { DELAY_TAG, RETURN_TAG, SET_ALIAS_TAG, PREPARE_TAG,
DESTROY_TAG, ABORT_TAG, SET_DATA_TAG, COPY_TAG, SET_MULTIPLEX_TAG,
SET_DEMULTIPLEX_TAG, NEW_TAG, DELETE_TAG, SET_GLOBAL_BEHAVIOUR_TAG, RUN_TAG,
STOP_TAG, SET_PARAMETERS_TAG, PLUG_TAG, SET_RGS_TAG, SET_OPACITY_TAG,
SET_PRESENTATION_PRIORITY_TAG, SET_VISIBLE_DURATION_TAG,
SET_TEMPORAL_TERMINATION_TAG, SET_CURRENT_TEMPORAL_POSITION_TAG,
SET_VISIBLE_DURATION_POSITION_TAG, SET_SPEED_TAG, SET_TIMESTONES_TAG,
SET_PERCEPTIBLE_SIZE_PROJECTION_TAG, SET_RESIZING_STRATEGY_TAG,
SET_ASPECT_RATIO_PRESERVED_TAG, SET_VISIBLE_SIZE_TAG,
SET_VISIBLE_SIZES_ADJUSTMENT_TAG, SET_BOX_TAG, SET_DEFAULT_BACKGROUND_TAG,
SET_ATTACHMENT_POINT_TAG, SET_ATTACHMENT_POINT_POSITION_TAG,
SET_VISIBLE_SIZES_ALIGNMENT_TAG, SET_MOVING_ABILITY_TAG,
SET_RESIZING_ABILITY_TAG, SET_SCALING_ABILITY_TAG, SET_SCROLLING_ABILITY_TAG,
SET_AUDIBLE_VOLUME_TAG, SET_AUDIBLE_COMPOSITION_EFFECT_TAG,
SET_STREAM_CHOICE_TAG, SET_SELECTABILITY_TAG, SET_SELECTION_STATUS_TAG,
SET_SELECTION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG, SET_MODIFIABILITY_TAG,
SET_MODIFICATION_STATUS_TAG,
SET_MODIFICATION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG, SET_BUTTON_STYLE_TAG,
SET_SLIDER_STYLE_TAG, SET_ENTRY_FIELD_STYLE_TAG, SET_MENU_STYLE_TAG,
SET_SCROLLING_LIST_STYLE_TAG, SET_NO_STYLE_TAG, NEW_CHANNEL_TAG,
DELETE_CHANNEL_TAG, SET_CHANNEL_PERCEPTABILITY_TAG };
union ElementaryAction
switch (ElementaryActionTag){
    case DELAY_TAG:
        Delay
            delay;
    case RETURN_TAG:
        Return
            return;
```

```
case SET_ALIAS_TAG:
    SetAlias
        set_alias;
case PREPARE_TAG:
    sequence<MhTargetsParameter,1>
        prepare;
case DESTROY_TAG:
    sequence<MhTargetsParameter,1>
        destroy;
case ABORT_TAG:
    sequence<MhTargetsParameter,1>
        abort;
case SET_DATA_TAG:
    SetData
        set_data;
case COPY_TAG:
    Copy
        copy;
case SET_MULTIPLEX_TAG:
    SetMultiplex
        set_multiplex;
case SET_DEMULTIPLEX_TAG:
    SetDemultiplex
        set_demultiplex;
case NEW_TAG:
    sequence<RtTargetsParameter,1>
        new;
case DELETE_TAG:
    sequence<RtTargetsParameter,1>
        delete;
case SET_GLOBAL_BEHAVIOUR_TAG:
    SetGlobalBehaviour
        set_global_behaviour;
case RUN_TAG:
    sequence<RtSocketTargetsParameter,1>
        run;
case STOP_TAG:
    sequence<RtSocketTargetsParameter,1>
        stop;
case SET_PARAMETERS_TAG:
    SetParameters
        set_parameters;
case PLUG_TAG:
    Plug
        plug;
case SET_RGS_TAG:
    SetRGS
        set_RGS;
case SET_OPACITY_TAG:
    SetOpacity
        set_opacity;
case SET_PRESENTATION_PRIORITY_TAG:
    SetPresentationPriority
        set_presentation_priority;
case SET_VISIBLE_DURATION_TAG:
    SetVisibleDuration
        set_visible_duration;
case SET_TEMPORAL_TERMINATION_TAG:
    SetTemporalTermination
        set_temporal_termination;
case SET_CURRENT_TEMPORAL_POSITION_TAG:
    SetCurrentTemporalPosition
        set_current_temporal_position;
case SET_VISIBLE_DURATION_POSITION_TAG:
    SetVisibleDurationPosition
        set_visible_duration_position;
case SET_SPEED_TAG:
    SetSpeed
        set_speed;
case SET_TIMESTONES_TAG:
    SetTimestones
        set_timestones;
case SET_PERCEPTIBLE_SIZE_PROJECTION_TAG:
    SetPerceptibleSizeProjection
        set_perceptible_size_projection;
case SET_RESIZING_STRATEGY_TAG:
    SetResizingStrategy
        set_resizing_strategy;
case SET_ASPECT_RATIO_PRESERVED_TAG:
    SetAspectRatioPreserved
        set_aspect_ratio_preserved;
case SET_VISIBLE_SIZE_TAG:
    SetVisibleSize
        set_visible_size;
case SET_VISIBLE_SIZES_ADJUSTMENT_TAG:
    SetVisibleSizesAdjustment
        set_visible_sizes_adjustment;
case SET_BOX_TAG:
    SetBox
        set_box;
case SET_DEFAULT_BACKGROUND_TAG:
    SetDefaultBackground
        set_default_background;
case SET_ATTACHMENT_POINT_TAG:
    SetAttachmentPoint
        set_attachment_point;
case SET_ATTACHMENT_POINT_POSITION_TAG:
    SetAttachmentPointPosition
        set_attachment_point_position;
case SET_VISIBLE_SIZES_ALIGNMENT_TAG:
    SetVisibleSizesAlignment
        set_visible_sizes_alignment;
case SET_MOVING_ABILITY_TAG:
    SetMovingAbility
        set_moving_ability;
case SET_RESIZING_ABILITY_TAG:
    SetResizingAbility
        set_resizing_ability;
case SET_SCALING_ABILITY_TAG:
    SetScalingAbility
        set_scaling_ability;
case SET_SCROLLING_ABILITY_TAG:
    SetScrollingAbility
        set_scrolling_ability;
case SET_AUDIBLE_VOLUME_TAG:
    SetAudibleVolume
        set_audible_volume;
case SET_AUDIBLE_COMPOSITION_EFFECT_TAG:
    SetAudibleCompositionEffect
        set_audible_composition_effect;
case SET_STREAM_CHOICE_TAG:
    SetStreamChoice
        set_stream_choice;
case SET_SELECTABILITY_TAG:
    SetSelectability
        set_selectability;
case SET_SELECTION_STATUS_TAG:
    SetSelectionStatus
        set_selection_status;
case SET_SELECTION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG:
    SetSelectionPresentationEffectResponsibility
        set_selection_presentation_effect_responsibility;
case SET_MODIFIABILITY_TAG:
    SetModifiability
        set_modifiability;
case SET_MODIFICATION_STATUS_TAG:
    SetModificationStatus
        set_modification_status;
case SET_MODIFICATION_PRESENTATION_EFFECT_RESPONSIBILITY_TAG:
    SetModificationPresentationEffectResponsibility
        set_modification_presentation_effect_responsibility;
case SET_BUTTON_STYLE_TAG:
    SetButtonStyle
        set_button_style;
case SET_SLIDER_STYLE_TAG:
    SetSliderStyle
        set_slider_style;
case SET_ENTRY_FIELD_STYLE_TAG:
    SetEntryFieldStyle
        set_entry_field_style;
case SET_MENU_STYLE_TAG:
    SetMenuStyle
        set_menu_style;
case SET_SCROLLING_LIST_STYLE_TAG:
    SetScrollingListStyle
        set_scrolling_list_style;
case SET_NO_STYLE_TAG:
    sequence<RtSocketTargetsParameter,1>
        set_no_style;
case NEW_CHANNEL_TAG:
    NewChannel
        new_channel;
case DELETE_CHANNEL_TAG:
    sequence<ChannelTargetsParameter,1>
        delete_channel;
case SET_CHANNEL_PERCEPTABILITY_TAG:
    SetChannelPerceptability
        set_channel_perceptability;
};

// Corresponding MHEG datatype: Keywords
//==========================================================================
typedef sequence<string> Keywords;

// Corresponding MHEG datatype: Description
//==========================================================================
struct Description {
sequence<string,1>
    name;
sequence<string,1>
    owner;
sequence<string,1>
    version;
sequence<string,1>
    date;
sequence<Keywords,1>
    keywords;
sequence<string,1>
    copyright;
sequence<string,1>
    licence;
sequence<string,1>
    comment;
};

// Corresponding MHEG datatype: Class-Identifier
//==========================================================================
enum ClassIdentifier {
ACTION_CLASS,
LINK_CLASS,
SCRIPT_CLASS,
CONTENT_CLASS,
MULTIPLEXED_CONTENT_CLASS,
COMPOSITE_CLASS,
CONTAINER_CLASS,
DESCRIPTOR_CLASS
};

// Corresponding MHEG datatype: OBJECTIDENTIFIER
//==========================================================================
struct OBJECTIDENTIFIER {
unsigned short root;
unsigned short arc1;
unsigned short arc2;
unsigned short arc3;
};
```

```
// Corresponding MHEG datatype: Synchro-Indicator
//=========================================================================
enum SynchroIndicator {
    SERIAL,
    PARALLEL
};

// Corresponding MHEG datatype: Synchro-Indicator-Macro
//=========================================================================
struct SynchroIndicatorMacro {
    MacroDefId
        macro_def_id;
    sequence<SynchroIndicator,1>
        synchro_indicator;
};

// Corresponding MHEG datatype: Synchro-Indicator-Parameter
//=========================================================================
enum SynchroIndicatorParameterTag { SYNCHRO_INDICATOR_TAG,
SYNCHRO_INDICATOR_MACRO_TAG };
union SynchroIndicatorParameter
switch (SynchroIndicatorParameterTag){
    case SYNCHRO_INDICATOR_TAG:
        SynchroIndicator
            synchro_indicator;
    case SYNCHRO_INDICATOR_MACRO_TAG:
        SynchroIndicatorMacro
            synchro_indicator_macro;
};

// Corresponding MHEG datatype: Performances
//=========================================================================
typedef long Performances;

// Corresponding MHEG datatype: Performances-Macro
//=========================================================================
struct PerformancesMacro {
    MacroDefId
        macro_def_id;
    sequence<Performances,1>
        performances;
};

// Corresponding MHEG datatype: Performances-Parameter
//=========================================================================
enum PerformancesParameterTag { PERFORMANCES_TAG, PERFORMANCES_MACRO_TAG };
union PerformancesParameter
switch (PerformancesParameterTag){
    case PERFORMANCES_TAG:
        Performances
            performances;
    case PERFORMANCES_MACRO_TAG:
        PerformancesMacro
            performances_macro;
};

interface ActionClass;

// Corresponding MHEG datatype: Action
//=========================================================================
enum ActionTag { ACTION_REFERENCE_TAG, ACTION_CLASS_TAG };
union Action
switch (ActionTag){
    case ACTION_REFERENCE_TAG:
        MhObjectReference
            action_reference;
    case ACTION_CLASS_TAG:
        ActionClass
            action_class;
};

// Corresponding MHEG datatype: Synchronized-Action
//=========================================================================
enum SynchronizedActionTag { ELEMENTARY_ACTION_TAG, ACTION_TAG };
```

```
union SynchronizedAction
switch (SynchronizedActionTag){
    case ELEMENTARY_ACTION_TAG:
        ElementaryAction
            elementary_action;
    case ACTION_TAG:
        Action
            action;
};

// Corresponding MHEG datatype: Action-Class
//==========================================================================
interface ActionClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute sequence<MHEGIdentifier,1>
        mheg_identifier;
attribute sequence<Description,1>
        the_description;
attribute sequence<SynchroIndicatorParameter,1>
        synchro_indicator_parameter;
attribute sequence<TargetsParameter,1>
        target_set_parameter;
attribute sequence<PerformancesParameter,1>
        performances_parameter;
attribute sequence<SynchronizedAction>
        synchronized_actions;
};

// Corresponding MHEG datatype: Logical-Operator
//==========================================================================
enum LogicalOperator {
    AND,
    OR,
    XOR,
    NAND,
    NOR,
    NXOR
};

// Corresponding MHEG datatype: Comparison-Operator
//==========================================================================
enum ComparisonOperator {
    EQUAL,
    NOT_EQUAL,
    GREATER,
    GREATER_EQUAL,
    LESS,
    LESS_EQUAL
};

// Corresponding MHEG datatype: Comparison-Value
//==========================================================================
enum ComparisonValueTag { COMPARISON_VALUE_GENERIC_VALUE_TAG,
COMPARISON_VALUE_COMPARISON_VALUE_CONSTANT_TAG };
union ComparisonValue
switch (ComparisonValueTag){
    case COMPARISON_VALUE_GENERIC_VALUE_TAG:
        GenericValue
            generic_value;
    case COMPARISON_VALUE_COMPARISON_VALUE_CONSTANT_TAG:
        ComparisonValueConstant
            comparison_value_constant;
};

// Corresponding MHEG datatype: Evaluated-Condition
//==========================================================================
struct EvaluatedCondition {
    ComparisonOperator
        comparison_operator;
    ComparisonValue
        comparison_value;
};
```

```
// Corresponding MHEG datatype: Previous-Condition
//==========================================================================
enum PreviousConditionTag { EVALUATED_CONDITION_TAG, NEGATION_TAG };
union PreviousCondition
switch (PreviousConditionTag){
    case EVALUATED_CONDITION_TAG:
        EvaluatedCondition
            evaluated_condition;
};

// Corresponding MHEG datatype: Generic-Condition
//==========================================================================
struct GenericCondition {
    EvaluatedValue
        source_value;
    sequence<PreviousCondition,1>
        previous_condition;
    EvaluatedCondition
        current_condition;
};

// Corresponding MHEG datatype: Link-Condition
//==========================================================================
enum LinkConditionTag { LOGICAL_COMBINATION_TAG, GENERIC_CONDITION_TAG };
union LinkCondition
switch (LinkConditionTag){
    case LOGICAL_COMBINATION_TAG:
        struct LogicalCombination {
            sequence<LogicalOperator,1>                   logical_operator;
            sequence<LinkCondition>                       conditions;
            } logical_combination;
    case GENERIC_CONDITION_TAG:
        GenericCondition
            generic_condition;
};

// Corresponding MHEG datatype: Macro-Resolution-Parameter
//==========================================================================
struct MacroResolutionParameter {
    MacroDefId
        macro_def_id;
    sequence<GenericValue,1>
        usage_value;
};

// Corresponding MHEG datatype: Link-Effect
//==========================================================================
struct LinkEffect {
    sequence<sequence<MacroResolutionParameter>,1>
        macro_resolution;
    Action
                        action;
};

// Corresponding MHEG datatype: Link-Class
//==========================================================================
interface LinkClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute sequence<MHEGIdentifier,1>
        mheg_identifier;
attribute sequence<Description,1>
        the_description;
attribute sequence<LinkCondition,1>
        link_condition;
attribute LinkEffect
            link_effect;
};

// Corresponding MHEG datatype: Script-Inclusion
//==========================================================================
enum ScriptInclusionTag { OCTETSTRING_TAG, BITSTRING_TAG };
union ScriptInclusion
```

```
switch (ScriptInclusionTag){
    case OCTETSTRING_TAG:
        string
            octetstring;
    case BITSTRING_TAG:
        string
            bitstring;
};

// Corresponding MHEG datatype: Script-Data
//==========================================================================
enum ScriptDataTag { SCRIPT_INCLUSION_TAG, DATA_REFERENCE_TAG };
union ScriptData
switch (ScriptDataTag){
    case SCRIPT_INCLUSION_TAG:
        ScriptInclusion
            script_inclusion;
    case DATA_REFERENCE_TAG:
        DataReference
            data_reference;
};

// Corresponding MHEG datatype: Script-Class
//==========================================================================
interface ScriptClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute sequence<MHEGIdentifier,1>
        mheg_identifier;
attribute sequence<Description,1>
        the_description;
attribute sequence<ScriptClassification,1>
        script_classification;
attribute ScriptHook
            script_hook_information;
attribute ScriptData
            script_data;
};

// Corresponding MHEG datatype: Data-Inclusion
//==========================================================================
enum DataInclusionTag { DATA_INCLUSION_OCTET_STRING_TAG,
DATA_INCLUSION_BIT_STRING_TAG };
union DataInclusion
switch (DataInclusionTag){
    case DATA_INCLUSION_OCTET_STRING_TAG:
        string
            octet_string;
    case DATA_INCLUSION_BIT_STRING_TAG:
        string
            bit_string;
};

// Corresponding MHEG datatype: Content-Data
//==========================================================================
enum ContentDataTag { CONTENT_DATA_DATA_INCLUSION_TAG,
CONTENT_DATA_DATA_REFERENCE_TAG };
union ContentData
switch (ContentDataTag){
    case CONTENT_DATA_DATA_INCLUSION_TAG:
        DataInclusion
            data_inclusion;
    case CONTENT_DATA_DATA_REFERENCE_TAG:
        DataReference
            data_reference;
};

// Corresponding MHEG datatype: Content-Class
//==========================================================================
interface ContentClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute MHEGIdentifier
        mheg_identifier;
attribute Description
            the_description;
attribute ContentClassification
        classification;
attribute OriginalPerception
        original_perception;
attribute ContentHook
            content_hook;
attribute ContentData
            content_data;
};

// Corresponding MHEG datatype: Multiplexed-Stream
//==========================================================================
struct MultiplexedStream {
    sequence<sequence<long>,1>
        stream_identifier;
    sequence<ContentClassification,1>
        stream_classification;
    sequence<OriginalPerception,1>
        stream_original_perception;
    ContentHook
                   hook_stream;
};

// Corresponding MHEG datatype: Multiplexed-Content-Class
//==========================================================================
interface MultiplexedContentClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute sequence<MHEGIdentifier,1>
        mheg_identifier;
attribute sequence<Description,1>
        the_description;
attribute sequence<ContentClassification,1>
        mux_classification;
attribute sequence<OriginalPerception,1>
        original_perception;
attribute ContentHook
            mux_hook;
attribute ContentData
            mux_data;
attribute sequence<sequence<MultiplexedStream>,1>
        multiplexed_streams;
};

// Corresponding MHEG datatype: Link
//==========================================================================
enum LinkTag { LINK_REFERENCE_TAG, LINK_CLASS_TAG };
union Link
switch (LinkTag){
    case LINK_REFERENCE_TAG:
        MhObjectReference
            mh_object_reference;
    case LINK_CLASS_TAG:
        LinkClass
            link;
};

// Corresponding MHEG datatype: Specific-Behaviour
//==========================================================================
enum SpecificBehaviourTag { ACTIONS_TAG, LINKS_TAG };
union SpecificBehaviour
switch (SpecificBehaviourTag){
    case ACTIONS_TAG:
        sequence<Action>
            actions;
    case LINKS_TAG:
        sequence<Link>
            links;
};
```

```
// Corresponding MHEG datatype: Availability-Start-Up
// Corresponding MHEG datatype: Availability-Close-Down
// Corresponding MHEG datatype: Rt-Availability-Start-Up
//==========================================================================
enum PredefinedBehaviourTag { PREDEFINED_BEHAVIOUR_LINK_CLASS_TAG,
PREDEFINED_BEHAVIOUR_INHIBIT_TAG };
union PredefinedBehaviour
switch (PredefinedBehaviourTag){
    case PREDEFINED_BEHAVIOUR_LINK_CLASS_TAG:
        LinkClass
            link_class;
};

// Corresponding MHEG datatype: Predefined-Behaviour
//==========================================================================
struct PredefinedBehaviours {
    sequence<PredefinedBehaviour,1>
        availability_start_up;
    sequence<PredefinedBehaviour,1>
        availability_close_down;
    sequence<PredefinedBehaviour,1>
        rt_availability_start_up;
    sequence<LinkClass,1>
        rt_availability_close_down;
};

// Corresponding MHEG datatype: Composition-Behaviour
//==========================================================================
struct CompositionBehaviour {
    PredefinedBehaviours
        predefined_behaviours;
    SpecificBehaviour
        specific_behaviour;
};

interface CompositeClass;

// Corresponding MHEG datatype: Component
//==========================================================================
enum ComponentTag { MH_OBJECT_REFERENCE_TAG, CONTENT_TAG,
MULTIPLEXED_CONTENT_TAG, COMPOSITE_TAG };
union Component
switch (ComponentTag){
    case MH_OBJECT_REFERENCE_TAG:
        MhObjectReference
            mh_object_reference;
    case CONTENT_TAG:
        ContentClass
            content;
    case MULTIPLEXED_CONTENT_TAG:
        MultiplexedContentClass
            multiplexed_content;
    case COMPOSITE_TAG:
        CompositeClass
            composite;
};

// Corresponding MHEG datatype: Associated-Model
//==========================================================================
enum AssociatedModelTag { ASSOCIATED_MODEL_COMPONENT_TAG,
ASSOCIATED_MODEL_ASSOCIATED_LABEL_TAG };
union AssociatedModel
switch (AssociatedModelTag){
    case ASSOCIATED_MODEL_COMPONENT_TAG:
        Component
            component;
    case ASSOCIATED_MODEL_ASSOCIATED_LABEL_TAG:
        string
            associated_label;
};
```

```
// Corresponding MHEG datatype: Element
//==========================================================================
struct Element {
    long
        element_index;
    AssociatedModel
        associated_model;
};

// Corresponding MHEG datatype: Composition
//==========================================================================
struct Composition {
    long
        nb_elements;
    sequence<Element>
        elements;
};

// Corresponding MHEG datatype: Composite-Class
//==========================================================================
interface CompositeClass {
attribute OBJECTIDENTIFIER
        object_identification;
attribute sequence<MHEGIdentifier,1>
        mheg_identifier;
attribute sequence<Description,1>
        the_description;
attribute sequence<CompositionBehaviour,1>
        composition_behaviour;
attribute sequence<OriginalPerception,1>
        original_perception;
attribute Composition
            elements;
};

// Corresponding MHEG datatype: Container-Default-Behaviour
//==========================================================================
struct ContainerDefaultBehaviour {
    sequence<PredefinedBehaviour,1>
        availability_start_up;
    sequence<PredefinedBehaviour,1>
        availability_close_down;
};

interface ContainerClass;
interface DescriptorClass;

// Corresponding MHEG datatype: Mh-Element
//==========================================================================
enum MhElementTag { MH_ELEMENT_MH_OBJECT_REFERENCE_TAG, MH_ELEMENT_ACTION_TAG,
MH_ELEMENT_LINK_TAG, MH_ELEMENT_SCRIPT_TAG, MH_ELEMENT_CONTENT_TAG,
MH_ELEMENT_MUX_CONTENT_TAG, MH_ELEMENT_COMPOSITE_TAG, MH_ELEMENT_CONTAINER_TAG,
MH_ELEMENT_DESCRIPTOR_TAG };
union MhElement
switch (MhElementTag){
    case MH_ELEMENT_MH_OBJECT_REFERENCE_TAG:
        MhObjectReference
            mh_object_reference;
    case MH_ELEMENT_ACTION_TAG:
        ActionClass
            action;
    case MH_ELEMENT_LINK_TAG:
        LinkClass
            link;
    case MH_ELEMENT_SCRIPT_TAG:
        ScriptClass
            script;
    case MH_ELEMENT_CONTENT_TAG:
        ContentClass
            content;
    case MH_ELEMENT_MUX_CONTENT_TAG:
        MultiplexedContentClass
            mux_content;
```

```
     case MH_ELEMENT_COMPOSITE_TAG:
          CompositeClass
               composite;
     case MH_ELEMENT_CONTAINER_TAG:
          ContainerClass
               container;
     case MH_ELEMENT_DESCRIPTOR_TAG:
          DescriptorClass
               descriptor;
};

// Corresponding MHEG datatype: Container-Class
//=========================================================================
interface ContainerClass {
attribute OBJECTIDENTIFIER
          object_identification;
attribute sequence<MHEGIdentifier,1>
          mheg_identifier;
attribute sequence<Description,1>
          the_description;
attribute ContainerDefaultBehaviour
          container_default_behaviour;
attribute sequence<MhElement>
          container_elements;
};

// Corresponding MHEG datatype: Delay-Bounds
//=========================================================================
struct DelayBounds {
     long
          lower_value;
     long
          upper_value;
};

// Corresponding MHEG datatype: Degradation
//=========================================================================
enum Degradation {
     GUARANTEED,
     BEST_EFFORT,
     NOT_GUARANTEED
};

// Corresponding MHEG datatype: Quality-Of-Service
//=========================================================================
struct QualityOfService {
     sequence<Degradation,1>
          degradation;
     sequence<boolean,1>
          reliability;
     sequence<DelayBounds,1>
          delay_bounds;
     sequence<long,1>
          jitter_bounds;
};

// Corresponding MHEG datatype: Script-Class-Information
//=========================================================================
struct ScriptClassInformation {
     sequence<ScriptClassification,1>
          script_classification;
     sequence<ScriptHook,1>
          script_hook;
};

// Corresponding MHEG datatype: Alternative-Object
//=========================================================================
struct AlternativeObject {
     MhObjectReference
          content_or_mux_object;
     sequence<ContentHook,1>
          alternative_hook;
     sequence<MhObjectReference,1>
          alternative_descriptor;
```

```
    sequence<string,1>
        alternative_keyword;
};

// Corresponding MHEG datatype: Content-Class-Information
//==========================================================================
struct ContentClassInformation {
    sequence<ContentClassification,1>
        content_classification;
    sequence<ContentHook,1>
        content_hook;
    sequence<sequence<AlternativeObject>,1>
        alternative_object;
};

// Corresponding MHEG datatype: Stream-Information
//==========================================================================
struct StreamInformation {
    sequence<StreamIdentifier,1>
        stream_identifier;
    sequence<ContentClassInformation,1>
        content_class_information;
};

// Corresponding MHEG datatype: Multiplexed-Content-Class-Information
//==========================================================================
struct MultiplexedContentClassInformation {
    sequence<ContentClassification,1>
        multiplexed_content_classification;
    sequence<ContentHook,1>
        multiplexed_content_hook;
    sequence<long,1>
        stream_number;
    sequence<sequence<StreamInformation>,1>
        stream_information;
    sequence<sequence<AlternativeObject>,1>
        alternative_object;
};

// Corresponding MHEG datatype: Class-Specific
//==========================================================================
enum ClassSpecificTag { SCRIPT_CLASS_INFORMATION_TAG,
CONTENT_CLASS_INFORMATION_TAG, MULTIPLEXED_CONTENT_CLASS_INFORMATION_TAG };
union ClassSpecific
switch (ClassSpecificTag){
    case SCRIPT_CLASS_INFORMATION_TAG:
        ScriptClassInformation
            script_class_information;
    case CONTENT_CLASS_INFORMATION_TAG:
        ContentClassInformation
            content_class_information;
    case MULTIPLEXED_CONTENT_CLASS_INFORMATION_TAG:
        MultiplexedContentClassInformation
            multiplexed_content_class_information;
};

// Corresponding MHEG datatype: Object-Information
//==========================================================================
struct ObjectInformation {
    sequence<long,1>
        object_size;
    sequence<ClassIdentifier,1>
        class_identifier;
    sequence<ClassSpecific,1>
        class_specific;
    sequence<QualityOfService,1>
        quality_of_service;
};

// Corresponding MHEG datatype: Related-Object
//==========================================================================
struct RelatedObject {
    MhObjectReference
        object_reference;
    sequence<ObjectInformation,1>
        object_information;
};

// Corresponding MHEG datatype: System-Readable-Material
//==========================================================================
enum SystemReadableMaterialTag { SYSTEM_READABLE_MATERIAL_BITSTRING_TAG,
SYSTEM_READABLE_MATERIAL_OCTETSTRING_TAG };
union SystemReadableMaterial
switch (SystemReadableMaterialTag){
    case SYSTEM_READABLE_MATERIAL_BITSTRING_TAG:
        string
            bitstring;
    case SYSTEM_READABLE_MATERIAL_OCTETSTRING_TAG:
        string
            octetstring;
};

// Corresponding MHEG datatype: General-Information
//==========================================================================
struct GeneralInformation {
    sequence<string,1>
        readme;
    sequence<SystemReadableMaterial,1>
        system_readable_material;
};

// Corresponding MHEG datatype: Media-Type
//==========================================================================
enum MediaType {
    AUDIBLE,
    LEFT_AUDIBLE,
    RIGHT_AUDIBLE,
    VISIBLE
};

// Corresponding MHEG datatype: Predefined-Selection-Mode
//==========================================================================
enum PredefinedSelectionMode {
    MOUSE_BUTTON_1_DOWN,
    MOUSE_BUTTON_2_DOWN,
    MOUSE_BUTTON_3_DOWN,
    MOUSE_BUTTONS_1_2_DOWN,
    MOUSE_BUTTONS_1_3_DOWN,
    MOUSE_BUTTONS_2_3_DOWN,
    MOUSE_BUTTON_1_UP,
    MOUSE_BUTTON_2_UP,
    MOUSE_BUTTON_3_UP,
    MOUSE_BUTTONS_1_2_UP,
    MOUSE_BUTTONS_1_3_UP,
    MOUSE_BUTTONS_2_3_UP,
    MOUSE_SINGLE_CLICK_ON_BUTTON_1,
    MOUSE_SINGLE_CLICK_ON_BUTTON_2,
    MOUSE_SINGLE_CLICK_ON_BUTTON_3,
    MOUSE_SINGLE_CLICK_ON_BUTTONS_12,
    MOUSE_SINGLE_CLICK_ON_BUTTONS_13,
    MOUSE_SINGLE_CLICK_ON_BUTTONS_23,
    MOUSE_DOUBLE_CLICK_ON_BUTTON_1,
    MOUSE_DOUBLE_CLICK_ON_BUTTON_2,
    MOUSE_DOUBLE_CLICK_ON_BUTTON_3,
    MOUSE_DOUBLE_CLICK_ON_BUTTONS_12,
    MOUSE_DOUBLE_CLICK_ON_BUTTONS_13,
    MOUSE_DOUBLE_CLICK_ON_BUTTONS_23,
    MOUSE_TRIPLE_CLICK_ON_BUTTON_1,
    MOUSE_TRIPLE_CLICK_ON_BUTTON_2,
    MOUSE_TRIPLE_CLICK_ON_BUTTON_3,
    MOUSE_TRIPLE_CLICK_ON_BUTTONS_12,
    MOUSE_TRIPLE_CLICK_ON_BUTTONS_13,
    MOUSE_TRIPLE_CLICK_ON_BUTTONS_23,
    MOUSE_ENTER,
    MOUSE_LEAVE,
    MOUSE_POINTER_MOTION,
    MOUSE_BUTTON_1_MOTION,
    MOUSE_BUTTON_2_MOTION,
    MOUSE_BUTTON_3_MOTION,
    MOUSE_BUTTONS_1_2_MOTION,
    MOUSE_BUTTONS_1_3_MOTION,
    MOUSE_BUTTONS_2_3_MOTION,
    KEY_PRESS,
    KEY_RELEASE
};

// Corresponding MHEG datatype: Physical-Selection-Mode
//==========================================================================
enum PhysicalSelectionModeTag { PRIVATE_MODE_TAG, PREDEFINED_TAG };
union PhysicalSelectionMode
switch (PhysicalSelectionModeTag){
    case PRIVATE_MODE_TAG:
        long
            private_mode;
    case PREDEFINED_TAG:
        PredefinedSelectionMode
            predefined;
};

// Corresponding MHEG datatype: Selection-Mode
//==========================================================================
struct SelectionMode {
    long
        logical_selection_mode;
    sequence<PhysicalSelectionMode,1>
        physical_selection_mode;
};

// Corresponding MHEG datatype: Predefined-Modification-Mode
//==========================================================================
enum PredefinedModificationMode {
    MEDIA_EDITOR
};

// Corresponding MHEG datatype: Physical-Modification-Mode
//==========================================================================
enum PhysicalModificationModeTag { PHYSICAL_MODIFICATION_MODE_NUMERIC_TAG,
PHYSICAL_MODIFICATION_MODE_PREDEFINED_TAG };
union PhysicalModificationMode
switch (PhysicalModificationModeTag){
    case PHYSICAL_MODIFICATION_MODE_NUMERIC_TAG:
        long
            numeric;
    case PHYSICAL_MODIFICATION_MODE_PREDEFINED_TAG:
        PredefinedModificationMode
            predefined;
};

// Corresponding MHEG datatype: Modification-Mode
//==========================================================================
struct ModificationMode {
    long
        logical_modification_mode;
    sequence<PhysicalModificationMode,1>
        physical_modification_mode;
};

// Corresponding MHEG datatype: Channel-Information
//==========================================================================
struct ChannelInformation {
    sequence<ChannelIdentifier,1>
        channel_id;
    sequence<long,1>
        x_min;
    sequence<long,1>
        x_max;
    sequence<long,1>
        y_min;
```

```
     sequence<long,1>
          y_max;
     sequence<long,1>
          z_min;
     sequence<long,1>
          z_max;
     sequence<long,1>
          x_granularity;
     sequence<long,1>
          y_granularity;
     sequence<long,1>
          z_granularity;
     sequence<long,1>
          t_granularity;
     sequence<long,1>
          f_min;
     sequence<long,1>
          f_max;
     sequence<long,1>
          audio_dynamic;
     sequence<MediaType,1>
          media_type;
     sequence<SelectionMode,1>
          selection_mode;
     sequence<ModificationMode,1>
          modification_mode;
};

// Corresponding MHEG datatype: Interaction-Style-Information
//==========================================================================
enum InteractionStyleInformation {
     BUTTON,
     SLIDER,
     ENTRY_FIELD,
     MENU,
     SCROLLING_LIST
};

// Corresponding MHEG datatype: Descriptor-Class
//==========================================================================
interface DescriptorClass {
attribute OBJECTIDENTIFIER
          object_identification;
attribute sequence<MHEGIdentifier,1>
          mheg_identifier;
attribute sequence<Description,1>
          the_description;
attribute sequence<sequence<RelatedObject>,1>
          related_objects;
attribute sequence<sequence<MhObjectReference>,1>
          other_descriptors;
attribute sequence<GeneralInformation,1>
          general_information;
attribute sequence<sequence<ChannelInformation>,1>
          channel_information;
attribute sequence<sequence<InteractionStyleInformation>,1>
          interaction_styles_information;
};

}; // end of module
```

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communication
Series Z	Programming languages