DRAFT

pr **ETS 300 777-3**

July 1996

EUROPEAN TELECOMMUNICATION STANDARD

Source: ETSI TC-TE

Reference: DE/TE-01057-3

ICS: 33.020

Key words: API, MHEG, multimedia, terminal

Terminal Equipment (TE); End-to-end protocols for multimedia information retrieval services; Part 3: Application Programmable Interface (API) for MHEG-5

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

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

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

Contents

Foreword 7

1 Scope 9

2 Normative references 9

3 Definitions and abbreviations 9
3.1 Definitions 9
3.2 Abbreviations 11

4 Conformance 11
4.1 Implementation conformance 11
4.1.1 Conformance requirements 11
4.1.2 Conformance documentation 12
4.2 Application conformance 12
4.2.1 Strictly Conforming Application 12
4.2.2 Conforming Application 12
4.3 Test Methods 13

5 General description 13

6 MHEG-5 API design principles 15
6.1 Requirements on the API 15
6.2 Portability 15
6.3 Generic ability 15
6.4 Implementability 15
6.5 Minimal resource constraints 16
6.6 Structure of the API 16
6.6.1 Interpretation 16
6.6.2 Access 16
6.7 Examples 16
6.7.1 Interpretation module 16
6.7.1.1 Audio interface 16
6.7.1.1.1 setVolume operation 17
6.7.1.1.2 IDL definition 17
6.7.1.2 IDL definition 17
6.7.2 Access module 18
6.7.2.1 IDL definition 18

7 Generic API definition framework 18
7.1 MHEG elements input to MHEG API definition process 18
7.2 IDL elements output by MHEG API definition process 18
7.3 Requirements on the MHEG API definition process 19
7.4 General structure of the MHEG API 19
7.5 IDL non-object datatype definition 19
7.5.1 Name mapping 19
7.5.1.1 Data types 19
7.5.1.2 Components 20
7.5.1.3 Values 20
7.5.2 Type mapping 20
7.5.2.1 INTEGER 20
7.5.2.2 BOOLEAN 21
7.5.2.3 OCTET STRING 21
7.5.2.4 ENUMERATED 21
7.5.2.5 SEQUENCE OF 21
7.5.2.6 CHOICE 21

7.5.2.7 SEQUENCE 22
7.5.3 Order of declarations 22
7.6 IDL interface definition 24
7.7 IDL attribute definition 24
7.7.1 MHEG interchanged attributes 24
7.7.2 MHEG internal attributes 25
7.8 IDL operation definition 25
7.8.1 Operations mapping MHEG elementary actions 25
7.8.2 Operations enabling the deletion of an interface instance 27
7.8.3 Operations to attach and detach an interface instance to a MHEG entity 27
7.9 IDL exception definition 27

8 The MHEG-5 API 28
8.1 Interpretation module 28
8.1.1 Root interface 28
8.1.1.1 getAvailabilityStatus operation 28
8.1.1.2 getRunningStatus operation 28
8.1.1.3 IDL description 29
8.1.2 Group interface 29
8.1.2.1 setCachePriority operation 29
8.1.2.2 IDL description 29
8.1.3 Application interface 29
8.1.3.1 storePersistent operation 30
8.1.3.2 readPersistent operation 30
8.1.3.3 launch operation 31
8.1.3.4 spawn operation 31
8.1.3.5 quit operation 31
8.1.3.6 lockScreen operation 32
8.1.3.7 unlockScreen operation 32
8.1.3.8 openConnection operation 32
8.1.3.9 closeConnection operation 33
8.1.3.10 getEngineSupport operation 33
8.1.3.11 IDL description 34
8.1.4 Scene interface 34
8.1.4.1 transitionTo operation 34
8.1.4.2 setTimer operation 35
8.1.4.3 sendEvent operation 35
8.1.4.4 setCursorShape operation 36
8.1.4.5 setCursorPosition operation 36
8.1.4.6 getCursorPosition operation 37
8.1.4.7 IDL description 37
8.1.5 Ingredient interface 38
8.1.5.1 setData operation 38
8.1.5.2 preload operation 38
8.1.5.3 unload operation 39
8.1.5.4 IDL description 39
8.1.6 Link interface 39
8.1.6.1 activate operation 39
8.1.6.2 deactivate operation 40
8.1.6.3 IDL description 40
8.1.7 Procedure interface 40
8.1.7.1 runSynchronous operation 40
8.1.7.2 runAsynchronous operation 41
8.1.7.3 stop operation 41
8.1.7.4 IDL description 42
8.1.8 Palette interface 42
8.1.8.1 IDL description 42
8.1.9 Font interface 42
8.1.9.1 IDL description 42
8.1.10 CursorShape interface 42
8.1.10.1 IDL description 42
8.1.11 Variable interface 42
8.1.11.1 setVariable operation 42

8.1.11.2 getVariable operation....43
8.1.11.3 IDL description....43
8.1.12 Presentable interface....43
8.1.12.1 setData operation....43
8.1.12.2 run operation....44
8.1.12.3 stop operation....44
8.1.12.4 IDL description....45
8.1.13 TokenManager interface....45
8.1.13.1 move operation....45
8.1.13.2 moveTo operation....45
8.1.13.3 IDL description....46
8.1.14 TokenGroup interface....46
8.1.14.1 callActionSlot operation....46
8.1.14.2 IDL description....47
8.1.15 TemplateGroup interface....47
8.1.15.1 selectItem operation....47
8.1.15.2 deselectItem operation....47
8.1.15.3 toggleItem operation....47
8.1.15.4 callTemplateAction operation....48
8.1.15.5 setState operation....48
8.1.15.6 getState operation....49
8.1.15.7 setItemContent operation....49
8.1.15.8 getItemContent operation....49
8.1.15.9 IDL description....50
8.1.16 List interface....50
8.1.16.1 plug operation....50
8.1.16.2 unplug operation....51
8.1.16.3 scroll operation....51
8.1.16.4 IDL description....52
8.1.17 Visible interface....52
8.1.17.1 setPosition operation....52
8.1.17.2 setBoxSize operation....53
8.1.17.3 bringToFront operation....53
8.1.17.4 sendToBack operation....54
8.1.17.5 putBefore operation....54
8.1.17.6 putBehind operation....54
8.1.17.7 IDL description....55
8.1.18 Bitmap interface....55
8.1.18.1 setBitmapPaletteRef operation....55
8.1.18.2 scaleBitmap operation....56
8.1.18.3 setTransparency operation....56
8.1.18.4 IDL description....57
8.1.19 LineArt interface....57
8.1.19.1 setLineWidth operation....57
8.1.19.2 setLineColour operation....58
8.1.19.3 setFillColour operation....58
8.1.19.4 setLineartPalette operation....58
8.1.19.5 IDL description....59
8.1.20 Rectangle interface....59
8.1.20.1 IDL description....59
8.1.21 Text interface....59
8.1.21.1 getTextData operation....59
8.1.21.2 setFontRef operation....60
8.1.21.3 setTextPaletteRef operation....60
8.1.21.4 IDL description....61
8.1.22 Stream interface....61
8.1.22.1 setCounterTrigger operation....61
8.1.22.2 setSpeed operation....62
8.1.22.3 setCounterPosition operation....62
8.1.22.4 IDL description....63
8.1.23 Audio interface....63
8.1.23.1 setVolume operation....63

8.1.23.2 IDL description 63
8.1.24 Video interface 64
8.1.24.1 scaleVideo operation 64
8.1.24.2 IDL description 64
8.1.25 RTGraphics interface 64
8.1.25.1 IDL description 64
8.1.26 Interactible interface 65
8.1.26.1 setInteractionStatus operation 65
8.1.26.2 setHighlightStatus operation 65
8.1.26.3 setInteractiblePalette operation 66
8.1.26.4 IDL description 66
8.1.27 Slider interface 66
8.1.27.1 step operation 66
8.1.27.2 setSliderValue operation 67
8.1.27.3 setPortion operation 67
8.1.27.4 getSliderValue operation 68
8.1.27.5 IDL description 68
8.1.28 EntryField interface 68
8.1.28.1 setOverwriteMode operation 68
8.1.28.2 getEntryPoint operation 69
8.1.28.3 setEntryPoint operation 69
8.1.28.4 IDL description 70
8.1.29 HyperText interface 70
8.1.29.1 IDL description 70
8.1.30 Button interface 70
8.1.30.1 select operation 70
8.1.30.2 deselect operation 70
8.1.30.3 IDL description 71
8.1.31 Hotspot interface 71
8.1.31.1 select operation 71
8.1.31.2 IDL description 71
8.1.32 PushButton interface 71
8.1.32.1 select operation 72
8.1.32.2 setLabel operation 72
8.1.32.3 IDL description 72
8.1.33 SwitchButton interface 73
8.1.33.1 getSelectionStatus operation 73
8.1.33.2 deselect operation 73
8.1.33.3 toggle operation 73
8.1.33.4 setLabel operation 74
8.1.33.5 IDL description 74
8.1.34 DL definition 75
8.2 Access module 85
8.2.1 DL definition 85

History 104

Foreword

This draft European Telecommunication Standard (ETS) has been produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI), and is now submitted for the Public Enquiry phase of the ETSI standards approval procedure.

This ETS consists of 4 parts as follows:

Part 1: "Coding of multimedia and hypermedia information for basic multimedia applications";

Part 2: "Use of Digital Storage Media Command and Control (DSM-CC) for basic multimedia applications "(DE/TE-01057-2);

Part 3: "Application Programmable Inteface (API) for MHEG-5";

Part 4: "Videotex Man Machine Interface (VEMMI) enhancements to support broadband multimedia information retrieval services" (DE/TE-01057-4).

Proposed transposition dates	
Date of latest announcement of this ETS (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

Blank page

1 Scope

This European Telecommunications Standard (ETS) 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 Multimedia and Hypermedia information coding Experts Group (MHEG)-5 engines for their control by applications.

MHEG part 5 (ISO/IEC 13522-5 [1]) is a generic standard, which specifies the coded representation of interchanged multimedia/hypermedia information objects (MHEG-5 objects) for base level applications. These so-called MHEG-5 objects are handled, interpreted and presented by MHEG-5 engines.

2 Normative references

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

[1] ISO/IEC DIS 13522-5 (1996): "Information technology - Coding of Multimedia and Hypermedia Information - Part 5: MHEG Subset for Base Level Implementation".

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

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

[4] ETR 225: "Terminal Equipment (TE); Application Programmable Interface (API) and script representation for MHEG; Requirements and framework".

[5] ISO/IEC 8824 (1990): "Information Technology -- Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)".

[6] ISO/IEC 8825 (1990): "Information Technology -- Open Systems Interconnection - Specification of Basic Encoding Rules (BER) for Abstract Syntax Notation One (ASN.1)".

[7] ITU-T Recommendation I.113 (1994): "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 (1990): "Vocabulary of switching and signaling terms".

[10] ISO/IEC 14750-1 (Working Draft): "CORBA IDL as an Interface Definition Language for ODP Systems".

[11] ISO/IEC JTC1 Directives (1995), Supplement 2 (1995); "Guidelines for JTC API Standardisation".

3 Definitions and abbreviations

3.1 Definitions

Due to the particular nature of this ETS, some of the words and expressions used in this ETS 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 apply, in decreasing order:

- ISO/IEC DIS 13522-5 [1] MHEG-5;
- any other standard part of ISO/IEC 13522 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 signaling terms".

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.

function family: Cluster of functional MHEG-5 API requirements consisting of functions with related semantics and applying to the same type of target.

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

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 (ITU-T Recommendation I.113 [7]).

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

MHEG-5 API: The API provided by an MHEG-5 engine to applications for the manipulation of MHEG-5 objects, as defined in this ETS.

MHEG-5 application: A piece of software which uses the MHEG-5 API. A MHEG-5 application is therefore a client of an MHEG-5 engine.

MHEG-5 engine: A process or a set of processes that interpret MHEG-5 objects encoded according to the encoding specifications of ISO/IEC DIS 13522-5 [1]: Abstract Syntax Notation One (ASN.1).

MHEG-5 entity: Any MHEG-5 object, content data, script data, channel or other construction identified or referred to in ISO/IEC DIS 13522-5 [1].

MHEG-5 interpretation service: The service takes interchanged MHEG-5 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.

MHEG-5 object handling service: This facility physically creates, handles and maintains intermediate data structures necessary to implement a client access of the MHEG-5 objects and their contents.

MHEG-5 object: A coded representation of an instance of an MHEG-5 object class.

MHEG-5 using application: An application which involves the interchange of MHEG-5 objects within itself or with another application.

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.

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

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

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

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.

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

(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.

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

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

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

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

3.2 Abbreviations

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

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CORBA	Common Object Request Broker Architecture
DAVIC	Digital Audio Visual Council
EBNF	Extended Backus Naur Form
IDL	Interface Definition Language (as defined in ISO/IEC 14750-1 [10])
JTC	Joint Technical Committee
MHEG	Multimedia and Hypermedia information coding Experts Group
VM	Virtual Machine

4 Conformance

An implementation of this ETS is an MHEG-5 engine implementation which provides client MHEG-5 applications with one or several language bindings of the abstract Application Programming Interface (API) defined in this ETS.

An application of this ETS is an MHEG-5 application which uses a language binding of the abstract API defined in this ETS to control the behaviour of an MHEG-5 engine.

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

4.1 Implementation conformance

4.1.1 Conformance requirements

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

1) the implementation shall support all required behaviour defined in this ETS;

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

3) the implementation may provide additional functions or facilities not required by this ETS 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 ETS 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 ETS 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 ETS. The conformance document shall be in two parts. The first part shall have the same structure as this ETS, 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 ETS 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 ETS 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 ETS 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 ETS 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 ETS and the language binding specification shall be present in the conformance document.

The phrases "shall document" or "shall be documented" in this ETS or in a language binding specification for this ETS 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 ETS 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 ETS, in the language binding specification and in the applicable language standards. Such an application shall accept a behaviour described in this ETS 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 ETS and the language binding specification.

4.2.2 Conforming Application

A Conforming Application of a language binding specification for this ETS is an application that differs from a Strictly Conforming Application in that it may use optional facilities described in this ETS, in the language binding specification and in the applicable language standards, as well as non-standard facilities

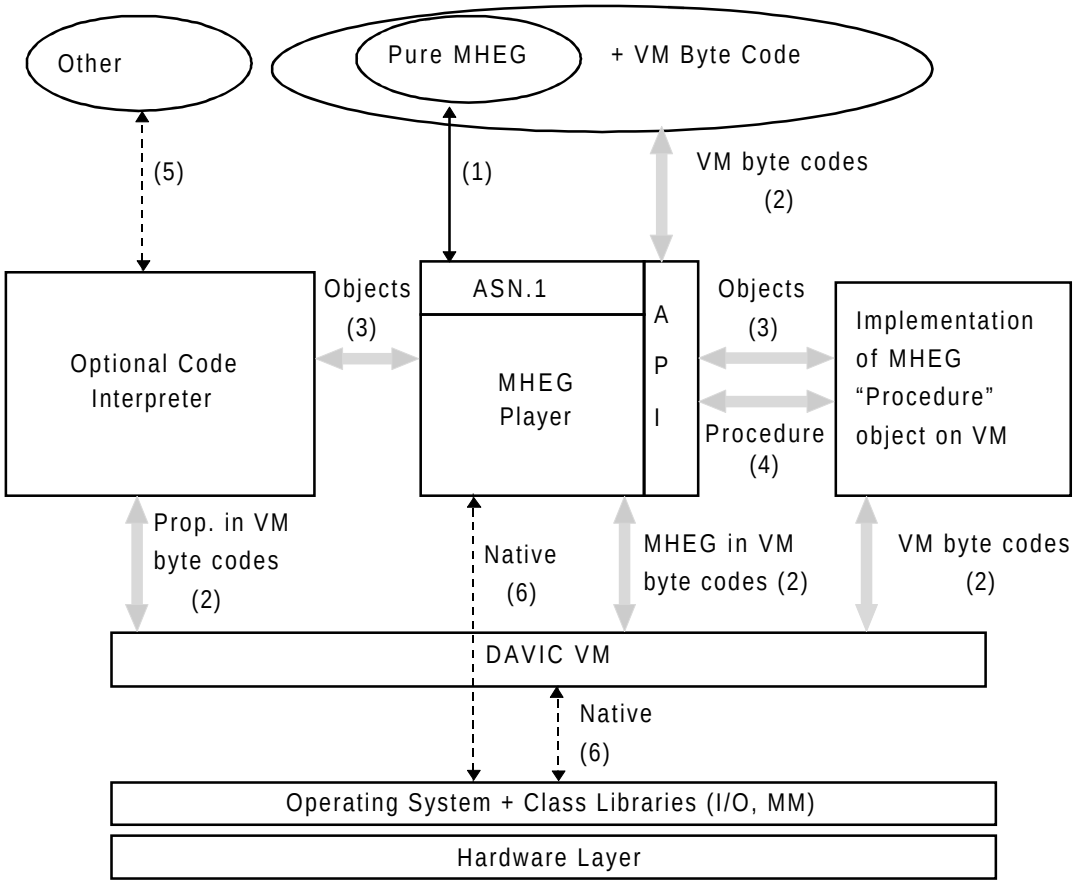
that are consistent with the ETS and with the language binding specification. Such an application shall fully document its requirements for these optional and extended facilities in addition to the documentation required of a Conforming Application.

4.3 Test Methods

Any measurement of conformance to a language binding specification for this ETS shall be performed using test methods that conform to ISO/IEC 9646 Parts 1 to 5 [2] and to any additional requirements that may be imposed by the language binding specification.

5 General description

This clause situates the MHEG-5 API within the architecture of an MHEG-5 using application. This is done making use of the DAVIC reference architecture. However, the architecture is compatible with the functional reference architecture for multimedia defined in ETR 173 [3]. The MHEG-5 API defined in this ETS is generic in the sense that it is applicable to other environments than DAVIC as well.



Legend: boxes represent functional modules;
arrows represent interfaces between functional modules;
ovals represent application data;
thin line arrows represent interfaces defined by DAVIC 1.0;
thick line arrows represent interfaces that are defined by DAVIC 1.1 in order to ensure application interoperability;
dashed line arrows represent interfaces that are not defined by DAVIC 1.1.

Figure 1: DAVIC STU reference architecture

The following main interfaces are identified in the architecture:

1) MHEG-5 objects;

2) DAVIC VM byte code;

3) MHEG-5 object manipulation from DAVIC VM byte code (MHEG-5 API);

4) MHEG-5 actions on script Procedure class (e.g. RunSynchronous, RunAsynchronous, parameter passing, etc.);

5) Other non-MHEG encoding;

6) Implementation of functional modules in native code.

The interface 1) is defined by DAVIC 1.0. The interfaces 2), 3), and 4) are defined by DAVIC 1.1. The interfaces 5) and 6) are not specified by DAVIC 1.1.

This ETS is specifying the MHEG-5 API at reference point (3).

6 MHEG-5 API design principles

This clause introduces the design principles that were used to design the MHEG-5 API. First the requirements for the API in the DAVIC environment are introduced and then the consequences of the requirements in terms of guidelines are developed.

An example given at the end of the clause shows how the derived guidelines are applied to develop the MHEG-5 API.

6.1 Requirements on the API

Following the recommendations of ETR 225 [4], ISO/IEC JTC1 Directives [11] and the requirements put forward by DAVIC, the MHEG API is defined as an abstract API specification, i.e. a language-independent description of the semantics of a functionality set in an abstract syntax using abstract data types.

In addition the following requirements apply:

- portability;
- generic ability;
- implementability;
- applicability on a minimal resource platform.

The following subclauses introduce the individual requirements in detail, highlights consequences of these requirements on the API-definition process and eventually show how the API definition meets the requirements.

6.2 Portability

The **portability** requirement states that the MHEG API standard should enable MHEG applications to use the MHEG object access and interpretation service provided by MHEG engines in a way independent of:

- the programming language used for the MHEG application;
- the underlying operating system.

This ETS meets the portability requirement by the definition of an abstract API specification making use of IDL.

6.3 Generic ability

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

This ETS meets the generic ability requirement through defining the MHEG API at the most basic level, e.g. by defining operations that match MHEG elementary actions and data types that match MHEG data types. This guarantees the range of MHEG object manipulations made available to applications is maximized.

6.4 Implementability

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 ETS meets the implementability requirement by the provision of using IDL for the definition of the API and by defining the API at the most basic level of abstraction. The use of IDL is beneficial because a number of tools are available for different platforms and operating systems to perform the language bindings of the given IDL interface definition. The choice of a basic level of abstraction (Interface operations map MHEG-5 elementary actions) guarantees that the MHEG API can be implemented without major difficulty. The MHEG-5 engine, in order to be MHEG-5 compliant, shall implement the elementary actions anyway. To make these actions available to using applications via an API only a simple mapping procedure is required. The API is therefore easy to implement.

6.5 Minimal resource constraints

The **minimal resource constraints** requirement states that the MHEG API standard should take into account constraints introduced by a minimal resource platform like a DAVIC compliant STU. The API should be well suited for such environments and hence be optimised in terms of use of STU resources.

This ETS meets the Minimal resource constraints requirement through defining the MHEG API at the most basic level, e.g. by defining operations that match MHEG elementary actions and data types that match MHEG data types. This guarantees results in a somewhat bulky API-definition because all MHEG-5 elementary actions including all relevant datatypes have to be mapped. An API definition on a higher level would be more compact in terms of API definition.

This advantage is more than compensated through the gains when implementing the API because the API operations can be directly mapped on MHEG-5 actions without additional computing.

6.6 Structure of the API

The API consists of the following two modules providing different API functionality:

- interpretation module;
- access module.

The modules are introduced in the following subclauses.

6.6.1 Interpretation

The interpretation module shall provide MHEG interpretation services. It is named "MHEG_5" module.

The following categories of operations shall be provided by the interfaces of the interpretation module:

- operations mapping MHEG elementary actions targeting the MHEG entity corresponding to the interface;

- operations enabling the deletion of an interface instance (see note);

- operations to attach and detach an interface instance to a MHEG entity.

NOTE: The creation of an interface instance is provided by the factory interface.

6.6.2 Access

The access module shall provide accessor and modifier services for encoded MHEG entities. It is named "MHEG_5_Access module".

6.7 Examples

6.7.1 Interpretation module

6.7.1.1 Audio interface

The `Audio` interface inherits from the `Presentable` interface.

The following subclauses define the operations provided by the `Audio` interface.

6.7.1.1.1 setVolume operation

Synopsis:

Interface: `Audio`

Operation: `setVolume`

Result: `void`

In: `GenericInteger new_volume`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setVolume` operation triggers the execution of the "SetVolume" elementary action with the attached `Audio` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.24.4.

The `new_volume` parameter corresponds to the "new_volume" parameter of the "SetVolume" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

6.7.1.1.2 IDL definition

```
interface Audio: Presentable
{
    void
        setVolume(
            in  GenericInteger  new_volume)
    raises(InvalidTarget, InvalidParameter);
};
```

6.7.1.2 IDL definition

```
module MHEG_5 {

// Corresponding MHEG datatype: ObjectReference
//=============================================================
struct ObjectReference {
    sequence<octet> group_identifier;
    long    object_number;
};

// Corresponding MHEG datatype: GenericInteger
//=============================================================
enum GenericIntegerTag { G_I_INTEGER_VALUE_TAG, G_I_INDIRECT_REFERENCE_TAG };
union GenericInteger
switch (GenericIntegerTag) {
    case G_I_INTEGER_VALUE_TAG:
        long    integer_value;
    case G_I_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Exceptions
//=============================================================================

exception InvalidTarget {};

enum CompletionStatus { YES, NO };

exception InvalidParameter {
        CompletionStatus        completion_status;
        unsigned short          rank;
};
```

```
interface Presentable {}; // to be defined

interface Audio: Presentable
{
    void
        setVolume(
            in  GenericInteger  new_volume)
    raises(InvalidTarget, InvalidParameter);
};

};
```

6.7.2 Access module

6.7.2.1 IDL definition

```
module MHEG_5 {

interface PresentableClass {}; // to be defined

// Corresponding MHEG datatype: AudioClass
//=============================================================
interface AudioClass : PresentableClass {
    attribute   sequence<long,1>    original_volume;
};

};
```

7 Generic API definition framework

Producing an MHEG API specification from a part of ISO/IEC 13522 [1] that describes presentation objects (hereafter called an MHEG specification) is a process that consists in producing IDL elements from MHEG elements.

The MHEG elements to which this process applies are described in subclause 7.1. The IDL elements to be produced from these MHEG elements are described in subclause 7.2. The rules used to produce the IDL elements from the MHEG elements are described in subclause 7.3.

7.1 MHEG elements input to MHEG API definition process

The different parts of ISO/IEC 13522 [1] share a number of key features. The following MHEG elements shall be present in the source MHEG specification:

- MHEG data types, described using ASN.1 or Extended Backus-Naur Form (EBNF);
- MHEG entities (i.e. objects targeted by MHEG elementary actions), related to each other by inheritance relationships;
- static and dynamic attributes of MHEG entities;
- MHEG elementary actions applying to MHEG entities;
- MHEG exceptions raised as the MHEG effect of elementary actions.

NOTE: The coded representation of MHEG-5 is defined using ISO/IEC 8824 [5] (ASN.1). The ASN.1 objects are encoded using ISO/IEC 8825 [6] (BER).

7.2 IDL elements output by MHEG API definition process

The API definition process should consist in mapping these elements to a set of IDL elements:

- IDL non-object types shall map MHEG data types;
- IDL object interfaces, related to each other by inheritance relationships, shall map MHEG entities;
- IDL attributes, provided by IDL object interfaces, shall map static and dynamic attributes of MHEG entities;

- IDL operations, provided by IDL object interfaces, shall map MHEG elementary actions;

- IDL exceptions shall map MHEG exceptions raised as the effect of elementary actions.

7.3 Requirements on the MHEG API definition process

The requirements on the MHEG API are introduced in clause 6.

7.4 General structure of the MHEG API

The MHEG API shall be defined using IDL as defined in ISO/IEC 14750-1 [10]. The use of IDL does not imply a Common Object Request Broker Architecture (CORBA) environment. Therefore, beside its use of IDL, the MHEG API shall be defined in a way independent of the CORBA architecture.

The IDL interface definition shall consist of two modules providing different API functionality. The first module shall provide MHEG interpretation services. It shall be named module `MHEG_<part>` where `<part>` is a number designating the part of ISO/IEC 13522 [1] targeted by the API. The second module shall provide accessor and modifier services for encoded MHEG entities. It shall be named module `MHEG_<part>_Access` where `<part>` is a number designating the part of ISO/IEC 13522 [1] targeted by the API.

The API shall provide an interface enabling the creation of an interface object instance. This interface shall be named `MHEGEntityManager`. Operations enabling the deletion of an IDL instance shall be provided by the interfaces that map MHEG entities.

Each interface mapping a MHEG entity that provides a common reference scheme for its subclasses shall also provide the following operations:

- `attach`;
- `detach`;
- `getIdentifier`.

The `attach` operation binds an interface object instance with an MHEG entity and returns the identifier of the bound object. The `detach` operation cancels the binding. The `getIdentifier` operation retrieves the identifier of the MHEG entity bound with the interface object instance.

The IDL concepts used to design the interface shall map MHEG concepts according to the rules described in the following subclauses. IDL identifiers shall map MHEG ASN.1 identifiers according to the rules defined in the following subclauses.

The following operations shall be synchronous: create, bind, unbind operations, accessors and modifiers, GET operations. Other elementary actions shall be mapped to asynchronous operations.

The API definition shall not use nested scoping levels.

7.5 IDL non-object datatype definition

IDL non-object types shall map MHEG datatypes expressed using ASN.1.

7.5.1 Name mapping

7.5.1.1 Data types

ASN.1 data type names shall be mapped to IDL type names as follows:

- if the ASN.1 type name consists of more than one word, no separators shall be used in the IDL name;

- each word shall start with a capital letter, all other letters shall be in lower case.

EXAMPLE: The ASN.1 `type-name` shall be mapped to the IDL `TypeName`.

7.5.1.2 Components

ASN.1 component names shall be mapped to IDL field names as follows:

- if the ASN.1 name consists of more than one word, the underscore character ("_") shall be used as a separator;

- all letters shall be in lower case;

- as IDL is not case-sensitive, type names and field names may collide when single-word names are used. In such a case, the field name shall be prefixed by "the_". The same rule shall be used when a single-word component name collides with an IDL keyword.

EXAMPLE:

```
// regular field name
TypeName    field_name

// collision between field name and type name
Alias       the_alias

// collision between field name and IDL keyword ("string")
Type        the_string
```

7.5.1.3 Values

ASN.1 value names shall be mapped to IDL value names as follows:

a) if the name consists of more than one word, the underscore character ("_") shall be used as a separator;

b) all letters shall be in upper case;

c) as an IDL enumeration does not create a new scope, values used within enumerations may collide. In such a case, for one of the colliding enumerations all value names (not only the one which collides) shall be prefixed by the name of the enumerated type. This type name shall be built using rules a) and b) (and not the regular rules for type names).

EXAMPLE:

```
// regular value name
FIRST_VALUE

// value name in case of a collision
FIRST_ENUMERATION_FIRST_VALUE
```

7.5.2 Type mapping

7.5.2.1 `INTEGER`

An ASN.1 `INTEGER` type shall be mapped to an IDL `long` type.

EXAMPLE:

```
-- ASN.1
Type ::= INTEGER

// IDL
typedef long Type;
```

7.5.2.2 BOOLEAN

An ASN.1 `BOOLEAN` type shall be mapped to an IDL `boolean` type.

EXAMPLE:

```
-- ASN.1
Type ::= BOOLEAN

// IDL
typedef boolean Type;
```

7.5.2.3 OCTET STRING

An ASN.1 `OCTET STRING` type shall be mapped to an IDL `sequence <octet>` type.

EXAMPLE:

```
-- ASN.1
Type ::= OCTET STRING

// IDL
typedef sequence <octet> Type;
```

7.5.2.4 ENUMERATED

An ASN.1 `ENUMERATED` type shall be mapped to an IDL `enum` type.

EXAMPLE:

```
-- ASN.1
Type ::= ENUMERATED {value_1 (1), value_2 (2)}

// IDL
enum Type (VALUE_1, VALUE_2);
```

7.5.2.5 SEQUENCE OF

An ASN.1 `SEQUENCE OF` type shall be mapped to an IDL `sequence` type.

EXAMPLE:

```
-- ASN.1
Type ::= SEQUENCE OF OtherType

// IDL
typedef sequence <OtherType> Type;
```

7.5.2.6 CHOICE

An ASN.1 `CHOICE` type shall be mapped to an IDL discriminated union by combining `enum` and `union`. The `enum` type name shall be derived from the name of the `CHOICE` type suffixed by "Tag" in order not to collide with the `union` type name. The `enum` value names shall be derived from the names of the `CHOICE` type fields suffixed by "_TAG" in order not to collide with field names within the `switch`. Within a `CHOICE` a `NULL` component shall be mapped on an empty `case`.

EXAMPLE:

```
-- ASN.1
Type ::= CHOICE
{
        a   A,
        b   B,
        c   NULL
}
```

```
// IDL
enum TypeTag { A_TAG, B_TAG, C_TAG };
union Type
switch TypeTag {
        case A_TAG:
            A   the_a;
        case B_TAG:
            B   the_b;
        // no‘case’ for C_TAG
};
```

7.5.2.7 SEQUENCE

An ASN.1 `SEQUENCE` type shall be mapped to an IDL `struct` type. `OPTIONAL` ASN.1 components with or without `DEFAULT` values shall be mapped to an IDL `sequence` of at most one element of that type.

EXAMPLE:

```
-- ASN.1
Type ::= SEQUENCE
{
        a   A OPTIONAL,
        b   B,
        c   INTEGER DEFAULT 0
}
```

```
// IDL
struct Type {
        sequence <A,1>  the_a;
        B           the_b;
        sequence <long,1>   c;
};
```

7.5.3 Order of declarations

The order of type declarations may be different between ASN.1 and IDL because IDL does not allow the use of a type before its declaration (and does not provide a forward declaration functionality). IDL type declarations shall be reordered to deal with this constraint.

To enable reordering, cross-references from the ASN.1 syntax shall be suppressed.

The following examples show cross-references commonly used in ASN.1 and how they should be removed.

EXAMPLE 1:

```
-- ASN.1 definition using a cross reference
-- Production rules used :
-- T = A | B
-- B = T C
-- Allowed values :
-- a c*

-- T refers to C
T ::= CHOICE
{
        a   A,
        b   B
}
-- B refers to T
B ::= SEQUENCE
{
        t   T,
        c   C
}

-- Equivalent ASN.1 definition without cross-reference
-- The information described remains the same, the structure has changed
-- Production rules used :
-- T = A C*

T ::= SEQUENCE
{
        a   A,
        s_o_c   SEQUENCE OF C OPTIONAL
}
```

EXAMPLE 2:

```
-- ASN.1 definition using a cross-reference
-- Production rules used :
-- T = A | B
-- B = C T
-- Allowed values :
-- c*a

-- T refers to C
T ::= CHOICE
{
        a   A,
        b   B
}
-- B refers to T
B ::= SEQUENCE
{
        c   C,
        t   T
}

-- Equivalent ASN.1 definition without cross-reference
-- The information described remains the same, the structure has changed
-- Production rules used :
-- T = C* A

T ::= SEQUENCE
{
        s_o_c   SEQUENCE OF C OPTIONAL,
        a   A
}
```

However, in most cases cross-referencing should be removed by the creation of nested types, as shown in example 3.

EXAMPLE 3:

```
-- ASN.1 definition using a cross-reference

-- T refers to A
T ::= CHOICE
{
        a   A,
        b   B
}

A ::= -- A definition that refers to T (either directly or indirectly)

-- Equivalent ASN.1 definition without cross-reference

T ::= CHOICE
{
        a   -- A definition
        b   B
}
```

The disadvantage of the method shown in example 3 is that very complex datatypes may be created. Moreover part of the datatypes so created might duplicate types already defined in the syntax.

Another option is to translate the "A" ASN.1 datatype into an "A" IDL interface embedding the corresponding "A" IDL datatype. This allows the use of a forward declaration of the "A" IDL interface as illustrated in example 4.

EXAMPLE 4:

```
interface A; // forward declaration

enum TTag { A_TAG, B_TAG };
union T
switch TTag {
        case A_TAG:
            A   the_a;
        case B_TAG:
            B   the_b;
};
```

```
interface A {

// this interface embeds the definition of type A which refers to type T

};
```

7.6 IDL interface definition

MHEG entities shall be mapped on IDL interfaces according to the following rules:

- every element that may be designated as a target of an MHEG elementary action shall be considered as an entity. Not all entities are explicitly defined by the MHEG standard. Moreover not all of them exist in the MHEG object model. Therefore the MHEG object model shall be refined to integrate all the entities used;

- if an elementary action may target different types of entities, a new entity shall be created aggregating these entities;

- the name of the new created entity shall be the concatenation of the names of the entities it aggregates using "Or" as a separator;

- the obtained object model is the API object model. There shall be a one-to-one correspondence between entities and interfaces;

- the IDL inheritance hierarchy shall correspond to the hierarchy of the API object model;

- interface names shall be derived from entity names using the same rules as for datatypes.

7.7 IDL attribute definition

The following categories of MHEG attributes are identified:

- interchanged attributes;
- internal attributes.

7.7.1 MHEG interchanged attributes

MHEG interchanged attributes shall be mapped to IDL attributes in the module `MHEG-<part>-access` according to the following rules:

a) every ASN.1 type whose name ends with "Class" shall be mapped to an interface;

b) within the type (see a)), the ASN.1 `COMPONENTS OF` ASN.1 shall be mapped to the IDL inheritance functionality;

c) within the type (see a)), every component shall be mapped to an attribute;

d) the naming and typing of attributes shall follow the rules defined in clause 6.

EXAMPLE:

```
-- ASN.1

B-Class ::= SEQUENCE
{
        COMPONENTS OF   A-Class,
        i       INTEGER,
        j       C-Type
}

// IDL

interface BClass : AClass {
        attribute long  i;
        attribute CType j;
};
```

7.7.2 MHEG internal attributes

MHEG internal attributes shall not be mapped to IDL attributes. MHEG internal attributes are accessed and modified through the use of MHEG elementary actions which shall be mapped to IDL operations.

7.8 IDL operation definition

The MHEG API interfaces shall provide the following categories of operations:

- operations that map MHEG elementary actions targeting the MHEG entity to which the interface corresponds;

- operations used to delete an interface instance (see note);

- operations used to attach and detach an interface instance to a MHEG entity.

NOTE: The creation of an interface instance is provided by the factory interface.

7.8.1 Operations mapping MHEG elementary actions

MHEG elementary actions shall be mapped to IDL operations according to the following rules:

a) there shall be a one-to-one correspondence between elementary actions and operations;

b) the interface that provides an operation shall correspond to the target entity of the elementary action;

c) the name of the operation shall be derived from the name of the elementary action using the following rules:

1) if an operation name consists of more then one word, no separators shall be used;
2) the operation name shall start with a lower-case letter;
3) each word (except the first) shall start with a capital letter.

d) a "Set" elementary action shall be mapped to an operation according to the following rules:

1) the return value of the operation shall be of type `void`;

2) there shall be only input parameters;

3) the datatypes and names of the parameters shall be derived from the ASN.1 elementary action definition using the rules defined in clause 6;

4) the first parameter of the ASN.1 elementary action definition shall not be mapped (it represents the target of the elementary action);

5) if the macro functionality is used for the definition of a parameter, the embedded datatype shall be used;

6) if the embedded datatype referenced in d5) is not explicitly defined it shall be created as outlined in example 1.

e) a "Get" elementary action shall be mapped to an operation according to the following rules:

1) the return value of the operation shall be the value evaluated by the elementary action;

2) the datatype of the return value does not exist as such in the MHEG ASN.1 syntax because elementary actions are evaluating generic value. It shall therefore be one of the following:

- a simple datatype;
- a complex datatype that is used in the corresponding "Set" elementary action;
- if a complex datatype is returned and no corresponding "Set" elementary action exists, the datatype shall be created.

3) there shall be no output parameters, all results shall be passed using the return value;

4) for input parameters, the rules defined in d3), d4), d5), d6) shall apply.

EXAMPLE:

```
Elementary-Action-N ::= SEQUENCE
{
        target      Target, -- to be skipped
        param1-param    Param1-Parameter, -- replace
        param2-param    Param2-Parameter  -- replace
}

Param1-Parameter ::= CHOICE
{
        param1      Param1,
        param1-macro    Param1-Macro
}

Param1-Macro ::=    SEQUENCE
{
        -- ...
        -- ...
}
Param1 ::=  -- ...

--      In "Elementary-Action-N" the "param1-parameter" parameter shall be replaced with the
--      "param1" parameter

Param2-Parameter ::= CHOICE
{
        a-param         A-Parameter,
        b-param         B-Parameter
}

A-Parameter ::= CHOICE
{
        a           A,
        a-macro     A-Macro
}

A-Macro ::= SEQUENCE
{
        -- ...
        -- ...
}

A ::=       -- ...

B-Parameter ::= SEQUENCE
{
        -- ...
        -- ...
}

B-Macro ::= SEQUENCE
{
        -- ...
        -- ...
}

B ::=       -- ...

--      In "Elementary-Action-N" the "param2-parameter" parameter cannot be replaced with the
--      "param2" parameter because it does not exist. Consequently an IDL datatype that
--      corresponds to the following ASN.1 datatype shall be created:

Param2 ::= CHOICE
{
        a       A,
        b       B
}
```

7.8.2 Operations enabling the deletion of an interface instance

Each interface shall provide an operation enabling the deletion of an interface instance. This operation shall have the following prototype:

```
void        kill();
```

7.8.3 Operations to attach and detach an interface instance to a MHEG entity

Each interface shall provide operations enabling attachment and detachment of an interface instance to a MHEG entity. These operations shall be named `attach` and `detach` respectively. The `attach` operation shall accept an MHEG entity reference as parameter, resolve this reference and return an MHEG entity identifier. The `detach` operation shall accept no parameter.

Each interface shall provide an operation to retrieve the identifier of the MHEG entity attached to an interface instance. This operation shall be named `getIdentifier`. It shall accept no parameter and return an MHEG entity identifier.

7.9 IDL exception definition

IDL exceptions shall map MHEG error conditions associated with elementary actions. Error conditions should be classified according to their nature so as to get a limited set of IDL exceptions.

The following set of exceptions is recommended:

- `InvalidTarget;`
- `InvalidParameter;`
- `NotBound;`
- `AlreadyBound`.

The `InvalidTarget` exception is raised when the targeted MHEG entity is not available. The `period` member returns the current state (life cycle 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 has been completed (with a default value assigned to the inadequate parameter) or not. The `parameter_number` member identifies the rank of the invalid parameter.

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

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.

The IDL definition of these exceptions is as follows:

```
exception InvalidTarget {
        unsigned short period;
};

enum CompletionStatus { YES, NO };

exception InvalidParameter {
        CompletionStatus    completion_status;
        unsigned short  parameter_number;
};

typedef long EntityIdentifier;

exception AlreadyBound {
        EntityIdentifier entity_identifier;
};

exception NotBound {};
```

The above mentioned exceptions shall be defined within the global scope.

If an error condition cannot be handled by the above mentioned exceptions, a specific exception shall be created at the appropriate interface scoping level.

The IDL exception names for specific exceptions shall be derived from the name of the MHEG error condition using the same rules as for datatypes:

a) no separators shall be used;

b) the exception name shall start with a capital letter;

c) if an exception name consists of more than one word, no separators between words shall be used and each word shall start with a capital letter.

8 The MHEG-5 API

The following subclauses specify the MHEG-5 API.

8.1 Interpretation module

8.1.1 `Root` interface

The following subclauses define the operations provided by the `Root` interface.

8.1.1.1 `getAvailabilityStatus` operation

Synopsis:

Interface: `Root`

Operation: `getAvailabilityStatus`

Result: `boolean`

Exception: `InvalidTarget`

Description:

The `getAvailabilityStatus` operation triggers the execution of the "GetAvailabilityStatus" elementary action with the attached `Root` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.2.4.

8.1.1.2 `getRunningStatus` operation

Synopsis:

Interface: `Root`

Operation: `getRunningStatus`

Result: `boolean`

Exception: `InvalidTarget`

Description:

The `getRunningStatus` operation triggers the execution of the "GetRunningStatus" elementary action with the attached `Root` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.2.4.

8.1.1.3 IDL description

```
interface Root
{
    boolean
        getAvailabilityStatus()
    raises(InvalidTarget);

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

8.1.2 `Group` interface

The `Group` interface inherits from the `Root` interface.

The following subclauses define the operations provided by the `Group` interface.

8.1.2.1 `setCachePriority` operation

Synopsis:

Interface: `Group`

Operation: `setCachePriority`

Result: `void`

In: `GenericInteger new_cache_priority`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setCachePriority` operation triggers the execution of the "SetCachePriority" elementary action with the attached `Group` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.3.4.

The `new_cache_priority` parameter corresponds to the "new_cache_priority" parameter of the "SetCachePriority" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.2.2 IDL description

```
interface Group: Root
{
    void
        setCachePriority(
            in  GenericInteger  new_cache_priority)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.3 `Application` interface

The `Application` interface inherits from the `Group` interface.

The following subclauses define the operations provided by the `Application` interface.

8.1.3.1 `storePersistent` operation

Synopsis:

Interface: `Application`

Operation: `storePersistent`

Result: `boolean`

In: `sequence<ObjectReference>` `in_variables`

In: `GenericOctetString` `out_file_name`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `storePersistent` operation triggers the execution of the "StorePersistent" elementary action with the attached `Application` object as its target.

The effect of the action on its target, the semantics of its parameters and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

The `in_variables` parameter corresponds to the "in_variables" parameter of the "StorePersistent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `out_file_name` parameter corresponds to the "out_file_name" parameter of the "StorePersistent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.3.2 `readPersistent` operation

Synopsis:

Interface: `Application`

Operation: `readPersistent`

Result: `boolean`

In: `sequence<ObjectReference>` `out_variables`

In: `GenericOctetString` `in_file_name`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `readPersistent` operation triggers the execution of the "ReadPersistent" elementary action with the attached `Application` object as its target.

The effect of the action on its target, the semantics of its parameters and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

The `out_variables` parameter corresponds to the "out_variables" parameter of the "ReadPersistent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `in_file_name` parameter corresponds to the "in_file_name" parameter of the "ReadPersistent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.3.3 `launch` operation

Synopsis:

Interface: `Application`

Operation: `launch`

Result: `void`

Exception: `InvalidTarget`

Description:

The `launch` operation triggers the execution of the "Launch" elementary action with the attached `Application` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.4 `spawn` operation

Synopsis:

Interface: `Application`

Operation: `spawn`

Result: `void`

Exception: `InvalidTarget`

Description:

The `spawn` operation triggers the execution of the "Spawn" elementary action with the attached `Application` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.5 `quit` operation

Synopsis:

Interface: `Application`

Operation: `quit`

Result: `void`

Exception: `InvalidTarget`

Description:

The `quit` operation triggers the execution of the "Quit" elementary action with the attached `Application` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.6 `lockScreen` operation

Synopsis:

Interface: `Application`

Operation: `lockScreen`

Result: `void`

Exception: `InvalidTarget`

Description:

The `lockScreen` operation triggers the execution of the "LockScreen" elementary action with the attached `Application` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.7 `unlockScreen` operation

Synopsis:

Interface: `Application`

Operation: `unlockScreen`

Result: `void`

Exception: `InvalidTarget`

Description:

The `unlockScreen` operation triggers the execution of the "UnlockScreen" elementary action with the attached `Application` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.8 `openConnection` operation

Synopsis:

Interface: `Application`

Operation: `openConnection`

Result: `boolean`

In: `GenericOctetString protocol`

In: `GenericOctetString address`

In: `GenericInteger connection_tag`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `openConnection` operation triggers the execution of the "OpenConnection" elementary action with the attached `Application` object as its target.

The effect of the action on its target, the semantics of its parameters and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

The `protocol` parameter corresponds to the "protocol" parameter of the "OpenConnection" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `address` parameter corresponds to the "address" parameter of the "OpenConnection" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `connection_tag` parameter corresponds to the "connection_tag" parameter of the "OpenConnection" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.3.9 `closeConnection` operation

Synopsis:

Interface: `Application`

Operation: `closeConnection`

Result: `void`

In: `GenericInteger connection_tag`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `closeConnection` operation triggers the execution of the "CloseConnection" elementary action with the attached `Application` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

The `connection_tag` parameter corresponds to the "connection_tag" parameter of the "CloseConnection" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.3.10 `getEngineSupport` operation

Synopsis:

Interface: `Application`

Operation: `getEngineSupport`

Result: `sequence<octet>`

Exception: `InvalidTarget`

Description:

The `getEngineSupport` operation triggers the execution of the "GetEngineSupport" elementary action with the attached `Application` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.4.4.

8.1.3.11 IDL description

```
interface Application: Group
{
    boolean
        storePersistent(
            in  sequence<ObjectReference>   in_variables,
            in  GenericOctetString  out_file_name)
    raises(InvalidTarget, InvalidParameter);

    boolean
        readPersistent(
            in  sequence<ObjectReference>   out_variables,
            in  GenericOctetString  in_file_name)
    raises(InvalidTarget, InvalidParameter);

    void
        launch()
    raises(InvalidTarget);

    void
        spawn()
    raises(InvalidTarget);

    void
        quit()
    raises(InvalidTarget);

    void
        lockScreen()
    raises(InvalidTarget);

    void
        unlockScreen()
    raises(InvalidTarget);

    boolean
        openConnection(
            in  GenericOctetString  protocol,
            in  GenericOctetString  address,
            in  GenericInteger  connection_tag)
    raises(InvalidTarget, InvalidParameter);

    void
        closeConnection(
            in  GenericInteger  connection_tag)
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getEngineSupport()
    raises(InvalidTarget);
};
```

8.1.4 Scene interface

The `Scene` interface inherits from the `Group` interface.

The following subclauses define the operations provided by the `Scene` interface.

8.1.4.1 `transitionTo` operation

Synopsis:

Interface: `Scene`

Operation: `transitionTo`

Result: `void`

In: `sequence<GenericInteger, 1>  connection_tag`

In: `GenericInteger  transition_effect`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `transitionTo` operation triggers the execution of the "TransitionTo" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

The `connection_tag` parameter corresponds to the "connection_tag" parameter of the "TransitionTo" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `transition_effect` parameter corresponds to the "transition_effect" parameter of the "TransitionTo" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.4.2 `setTimer` operation

Synopsis:

Interface: `Scene`

Operation: `setTimer`

Result: `void`

In: `GenericInteger timer_id`

In: `sequence<GenericInteger, 1> timer_value`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setTimer` operation triggers the execution of the "SetTimer" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

The `timer_id` parameter corresponds to the "timer_id" parameter of the "SetTimer" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `timer_value` parameter corresponds to the "timer_value" parameter of the "SetTimer" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.4.3 `sendEvent` operation

Synopsis:

Interface: `Scene`

Operation: `sendEvent`

Result: `void`

In: `ObjectReference event_source`

In: `EventType event_type`

In: `sequence<EmulatedEventData, 1>  emulated_event_data`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `sendEvent` operation triggers the execution of the "SendEvent" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

The `event_source` parameter corresponds to the "event_source" parameter of the "SendEvent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `event_type` parameter corresponds to the "event_type" parameter of the "SendEvent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `emulated_event_data` parameter corresponds to the "emulated_event_data" parameter of the "SendEvent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.4.4 `setCursorShape` operation

Synopsis:

Interface: `Scene`

Operation: `setCursorShape`

Result: `void`

In: `sequence<GenericObjectReference, 1> new_cursor_shape`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setCursorShape` operation triggers the execution of the "SetCursorShape" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

The `new_cursor_shape` parameter corresponds to the "new_cursor_shape" parameter of the "SetCursorShape" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.4.5 `setCursorPosition` operation

Synopsis:

Interface: `Scene`

Operation: `setCursorPosition`

Result: `void`

In: `GenericInteger  x_cursor`

In: `GenericInteger y_cursor`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setCursorPosition` operation triggers the execution of the "SetCursorPosition" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

The `x_cursor` parameter corresponds to the "x_cursor" parameter of the "SetCursorPosition" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `y_cursor` parameter corresponds to the "y_cursor" parameter of the "SetCursorPosition" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.4.6 `getCursorPosition` operation

Synopsis:

Interface: `Scene`

Operation: `getCursorPosition`

Result: `CursorPosition`

Exception: `InvalidTarget`

Description:

The `getCursorPosition` operation triggers the execution of the "GetCursorPosition" elementary action with the attached `Scene` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.5.4.

8.1.4.7 IDL description

```
interface Scene: Group
{
    void
        transitionTo(
            in  sequence<GenericInteger, 1> connection_tag,
            in  GenericInteger  transition_effect)
    raises(InvalidTarget, InvalidParameter);

    void
        setTimer(
            in  GenericInteger  timer_id,
            in  sequence<GenericInteger, 1> timer_value)
    raises(InvalidTarget, InvalidParameter);

    void
        sendEvent(
            in  ObjectReference event_source,
            in  EventType   event_type,
            in  sequence<EmulatedEventData, 1>  emulated_event_data)
    raises(InvalidTarget, InvalidParameter);

    void
        setCursorShape(
            in  sequence<GenericObjectReference, 1> new_cursor_shape)
    raises(InvalidTarget, InvalidParameter);
```

```
    void
        setCursorPosition(
            in  GenericInteger  x_cursor,
            in  GenericInteger  y_cursor)
    raises(InvalidTarget, InvalidParameter);

    CursorPosition
        getCursorPosition()
    raises(InvalidTarget);
};
```

8.1.5 `Ingredient` interface

The `Ingredient` interface inherits from the `Root` interface.

The following subclauses define the operations provided by the `Ingredient` interface.

8.1.5.1 `setData` operation

Synopsis:

Interface: `Ingredient`

Operation: `setData`

Result: `void`

In: `NewContent  new_content`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setData` operation triggers the execution of the "SetData" elementary action with the attached `Ingredient` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.6.4.

The `new_content` parameter corresponds to the "new_content" parameter of the "SetData" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.5.2 `preload` operation

Synopsis:

Interface: `Ingredient`

Operation: `preload`

Result: `void`

Exception: `InvalidTarget`

Description:

The `preload` operation triggers the execution of the "Preload" elementary action with the attached `Ingredient` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.6.4.

8.1.5.3 `unload` operation

Synopsis:

Interface: `Ingredient`

Operation: `unload`

Result: `void`

Exception: `InvalidTarget`

Description:

The `unload` operation triggers the execution of the "Unload" elementary action with the attached `Ingredient` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.6.4.

8.1.5.4 IDL description

```
interface Ingredient: Root
{
    void
        setData(
            in  NewContent  new_content)
    raises(InvalidTarget, InvalidParameter);

    void
        preload()
    raises(InvalidTarget);

    void
        unload()
    raises(InvalidTarget);
};
```

8.1.6 `Link` interface

The `Link` interface inherits from the `Ingredient` interface.

The following subclauses define the operations provided by the `Link` interface.

8.1.6.1 `activate` operation

Synopsis:

Interface: `Link`

Operation: `activate`

Result: `void`

Exception: `InvalidTarget`

Description:

The `activate` operation triggers the execution of the "Activate" elementary action with the attached `Link` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.7.4.

8.1.6.2 `deactivate` operation

Synopsis:

Interface: `Link`

Operation: `deactivate`

Result: `void`

Exception: `InvalidTarget`

Description:

The `deactivate` operation triggers the execution of the "Deactivate" elementary action with the attached `Link` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.7.4.

8.1.6.3 IDL description

```
interface Link: Ingredient
{
    void
        activate()
    raises(InvalidTarget);

    void
        deactivate()
    raises(InvalidTarget);
};
```

8.1.7 `Procedure` interface

The `Procedure` interface inherits from the `Ingredient` interface.

The following subclauses define the operations provided by the `Procedure` interface.

8.1.7.1 `runSynchronous` operation

Synopsis:

Interface: `Procedure`

Operation: `runSynchronous`

Result: `boolean`

In: `Parameters in_parameters`

Out: `Parameters out_parameters`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `runSynchronous` operation triggers the execution of the "RunSynchronous" elementary action with the attached `Procedure` object as its target.

The effect of the action on its target, the semantics of its parameters and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.8.4.

The `in_parameters` parameter corresponds to the "in_parameters" parameter of the "RunSynchronous" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `out_parameters` parameter corresponds to the "out_parameters" parameter of the "RunSynchronous" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.7.2 `runAsynchronous` operation

Synopsis:

Interface: `Procedure`

Operation: `runAsynchronous`

Result: `boolean`

In: `Parameters in_parameters`

Out: `Parameters out_parameters`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `runAsynchronous` operation triggers the execution of the "RunAsynchronous" elementary action with the attached `Procedure` object as its target.

The effect of the action on its target, the semantics of its parameters and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.8.4.

The `in_parameters` parameter corresponds to the "in_parameters" parameter of the "RunAsynchronous" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `out_parameters` parameter corresponds to the "out_parameters" parameter of the "RunAsynchronous" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.7.3 `stop` operation

Synopsis:

Interface: `Procedure`

Operation: `stop`

Result: `void`

Exception: `InvalidTarget`

Description:

The `stop` operation triggers the execution of the "Stop" elementary action with the attached `Procedure` object as its target.

The effect of the action on its target are defined by ISO/IEC DIS 13522-5 [1], subclause 2.8.4.

8.1.7.4 IDL description

```
interface Procedure: Ingredient
{
    boolean
        runSynchronous(
            in  Parameters  in_parameters)
    raises(InvalidTarget, InvalidParameter);

    boolean
        runAsynchronous(
            in  Parameters  in_parameters)
    raises(InvalidTarget, InvalidParameter);

    void
        stop()
    raises(InvalidTarget);
};
```

8.1.8 `Palette` interface

The `Palette` interface inherits from the `Ingredient` interface.

The `Palette` interface does not define any specific operation.

8.1.8.1 IDL description

```
interface Palette: Ingredient
{};
```

8.1.9 `Font` interface

The `Font` interface inherits from the `Ingredient` interface.

The `Font` interface does not define any specific operation.

8.1.9.1 IDL description

```
interface Font: Ingredient
{};
```

8.1.10 `CursorShape` interface

The `CursorShape` interface inherits from the `Ingredient` interface.

The `CursorShape` interface does not define any specific operation.

8.1.10.1 IDL description

```
interface CursorShape: Ingredient
{};
```

8.1.11 `Variable` interface

The `Variable` interface inherits from the `Ingredient` interface.

The following subclauses define the operations provided by the `Variable` interface.

8.1.11.1 `setVariable` operation

Synopsis:

Interface: `Variable`

Operation: `setVariable`

Result: `void`

In: `NewVariableValue new_variable_value`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setVariable` operation triggers the execution of the "SetVariable" elementary action with the attached `Variable` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.12.4.

The `new_variable_value` parameter corresponds to the "new_variable_value" parameter of the "SetVariable" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.11.2 `getVariable` operation

Synopsis:

Interface: `Variable`

Operation: `getVariable`

Result: `VariableValue`

Exception: `InvalidTarget`

Description:

The `getVariable` operation retrieves the value of the attached `Variable` object.

8.1.11.3 IDL description

```
interface Variable: Ingredient
{
    void
        setVariable(
            in  NewVariableValue    new_variable_value)
    raises(InvalidTarget, InvalidParameter);

    VariableValue
        getVariable()
    raises(InvalidTarget);
};
```

8.1.12 `Presentable` interface

The `Presentable` interface inherits from the `Ingredient` interface.

The following subclauses define the operations provided by the `Presentable` interface.

8.1.12.1 `setData` operation

Synopsis:

Interface: `Presentable`

Operation: `setData`

Result: `void`

In: `NewContent new_content`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setData` operation triggers the execution of the "SetData" elementary action with the attached `Presentable` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.13.4.

The `new_content` parameter corresponds to the "new_content" parameter of the "SetData" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

Note that this operation does not appear in the interface IDL definition as it overrides the semantics of the inherited operation provided by the `Ingredient` interface.

8.1.12.2 `run` operation

Synopsis:

Interface: `Presentable`

Operation: `run`

Result: `void`

Exception: `InvalidTarget`

Description:

The `run` operation triggers the execution of the "Run" elementary action with the attached `Presentable` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.13.4.

8.1.12.3 `stop` operation

Synopsis:

Interface: `Presentable`

Operation: `stop`

Result: `void`

Exception: `InvalidTarget`

Description:

The `stop` operation triggers the execution of the "Stop" elementary action with the attached `Presentable` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.13.4.

8.1.12.4 IDL description

```
interface Presentable: Ingredient
{
    // the ‘setData’ operation is defined in the ‘Ingredient’ interface

    void
        run()
    raises(InvalidTarget);

    void
        stop()
    raises(InvalidTarget);
};
```

8.1.13 `TokenManager` interface

The following subclauses define the operations provided by the `TokenManager` interface.

8.1.13.1 `move` operation

Synopsis:

Interface: `TokenManager`

Operation: `move`

Result: `void`

In: `GenericInteger movement_identifier`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `move` operation triggers the execution of the "Move" elementary action with the attached `TokenManager` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.14.4.

The `movement_identifier` parameter corresponds to the "movement_identifier" parameter of the "Move" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.13.2 `moveTo` operation

Synopsis:

Interface: `TokenManager`

Operation: `moveTo`

Result: `void`

In: `GenericInteger index`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `moveTo` operation triggers the execution of the "MoveTo" elementary action with the attached `TokenManager` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.14.4.

The `index` parameter corresponds to the "index" parameter of the "MoveTo" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.13.3 IDL description

```
interface TokenManager
{
    void
        move(
            in  GenericInteger  movement_identifier)
    raises(InvalidTarget, InvalidParameter);

    void
        moveTo(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.14 `TokenGroup` interface

The `TokenGroup` interface inherits from the `Presentable` and the `TokenManager` interfaces.

The following subclauses define the operations provided by the `TokenGroup` interface.

8.1.14.1 `callActionSlot` operation

Synopsis:

Interface: `TokenGroup`

Operation: `callActionSlot`

Result: `void`

In: `GenericInteger index`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `callActionSlot` operation triggers the execution of the "CallActionSlot" elementary action with the attached `TokenGroup` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.15.4.

The `index` parameter corresponds to the "index" parameter of the "CallActionSlot" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.14.2 IDL description

```
interface TokenGroup: Presentable, TokenManager
{
    void
        callActionSlot(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.15 `TemplateGroup` interface

The `TemplateGroup` interface inherits from the `Presentable` and the `TokenManager` interfaces.

The following subclauses define the operations provided by the `TemplateGroup` interface.

8.1.15.1 `selectItem` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `selectItem`

Result: `void`

Exception: `InvalidTarget`

Description:

The `selectItem` operation triggers the execution of the "SelectItem" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

8.1.15.2 `deselectItem` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `deselectItem`

Result: `void`

Exception: `InvalidTarget`

Description:

The `deselectItem` operation triggers the execution of the "DeselectItem" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

8.1.15.3 `toggleItem` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `toggleItem`

Result: `void`

Exception: `InvalidTarget`

Description:

The `toggleItem` operation triggers the execution of the "ToggleItem" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

8.1.15.4 `callTemplateAction` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `callTemplateAction`

Result: `void`

In: `GenericInteger index`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `callTemplateAction` operation triggers the execution of the "CallTemplateAction" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

The `index` parameter corresponds to the "index" parameter of the "CallTemplateAction" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.15.5 `setState` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `setState`

Result: `void`

In: `GenericOctetString new_state`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setState` operation triggers the execution of the "SetState" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

The `new_state` parameter corresponds to the "new_state" parameter of the "SetState" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.15.6 `getState` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `getState`

Result: `sequence<octet>`

Exception: `InvalidTarget`

Description:

The `getState` operation triggers the execution of the "GetState" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

8.1.15.7 `setItemContent` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `setItemContent`

Result: `void`

In: `GenericOctetString new_items_content`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setItemContent` operation triggers the execution of the "SetItemContent" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

The `new_items_content` parameter corresponds to the "new_items_content" parameter of the "SetItemContent" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.15.8 `getItemContent` operation

Synopsis:

Interface: `TemplateGroup`

Operation: `getItemContent`

Result: `sequence<octet>`

Exception: `InvalidTarget`

Description:

The `getItemContent` operation triggers the execution of the "GetItemContent" elementary action with the attached `TemplateGroup` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.16.4.

8.1.15.9 IDL description

```
interface TemplateGroup: Presentable, TokenManager
{
    void
        selectItem()
    raises(InvalidTarget);

    void
        deselectItem()
    raises(InvalidTarget);

    void
        toggleItem()
    raises(InvalidTarget);

    void
        callTemplateAction(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);

    void
        setState(
            in  GenericOctetString  new_state)
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getState()
    raises(InvalidTarget);

    void
        setItemContent(
            in  GenericOctetString  new_items_content)
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getItemContent()
    raises(InvalidTarget);
};
```

8.1.16 `List` interface

The `List` interface inherits from the `TemplateGroup` interface.

The following subclauses define the operations provided by the `List` interface.

8.1.16.1 `plug` operation

Synopsis:

Interface: `List`

Operation: `plug`

Result: `void`

In: `NewItemData item_data`

In: `sequence<GenericInteger, 1>  index`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `plug` operation triggers the execution of the "Plug" elementary action with the attached `List` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.17.4.

The `item_data` parameter corresponds to the "item_data" parameter of the "Plug" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `index` parameter corresponds to the "index" parameter of the "Plug" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.16.2 `unplug` operation

Synopsis:

Interface: `List`

Operation: `unplug`

Result: `void`

In: `sequence<GenericInteger, 1>  index`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `unplug` operation triggers the execution of the "Unplug" elementary action with the attached `List` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.17.4.

The `index` parameter corresponds to the "index" parameter of the "Unplug" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.16.3 `scroll` operation

Synopsis:

Interface: `List`

Operation: `scroll`

Result: `void`

In: `ScrollType  scroll_type`

In: `GenericInteger  scroll_number`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `scroll` operation triggers the execution of the "Scroll" elementary action with the attached `List` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.17.4.

The `scroll_type` parameter corresponds to the "scroll_type" parameter of the "Scroll" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `scroll_number` parameter corresponds to the "scroll_number" parameter of the "Scroll" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.16.4 IDL description

```
interface List: TemplateGroup
{
    void
        plug(
            in  NewItemData item_data,
            in  sequence<GenericInteger, 1> index)
    raises(InvalidTarget, InvalidParameter);

    void
        unplug(
            in  sequence<GenericInteger, 1> index)
    raises(InvalidTarget, InvalidParameter);

    void
        scroll(
            in  ScrollType  scroll_type,
            in  GenericInteger  scroll_number)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.17 `Visible` interface

The `Visible` interface inherits from the `Presentable` interface.

The following subclauses define the operations provided by the `Visible` interface.

8.1.17.1 `setPosition` operation

Synopsis:

Interface: `Visible`

Operation: `setPosition`

Result: `void`

In: `GenericInteger` `new_x_position`

In: `GenericInteger` `new_y_position`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setPosition` operation triggers the execution of the "SetPosition" elementary action with the attached `Visible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

The new_x_position parameter corresponds to the "new_x_position" parameter of the "SetPosition" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The new_y_position parameter corresponds to the "new_y_position" parameter of the "SetPosition" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.17.2 setBoxSize operation

Synopsis:

Interface: Visible

Operation: setBoxSize

Result: void

In: GenericInteger x_box_size

In: GenericInteger y_box_size

Exception: InvalidTarget

Exception: InvalidParameter

Description:

The setBoxSize operation triggers the execution of the "SetBoxSize" elementary action with the attached Visible object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

The x_box_size parameter corresponds to the "x_box_size" parameter of the "SetBoxSize" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The y_box_size parameter corresponds to the "y_box_size" parameter of the "SetBoxSize" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.17.3 bringToFront operation

Synopsis:

Interface: Visible

Operation: bringToFront

Result: void

Exception: InvalidTarget

Description:

The bringToFront operation triggers the execution of the "BringToFront" elementary action with the attached Visible object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

8.1.17.4 `sendToBack` operation

Synopsis:

Interface: `Visible`

Operation: `sendToBack`

Result: `void`

Exception: `InvalidTarget`

Description:

The `sendToBack` operation triggers the execution of the "SendToBack" elementary action with the attached `Visible` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

8.1.17.5 `putBefore` operation

Synopsis:

Interface: `Visible`

Operation: `putBefore`

Result: `void`

In: `GenericObjectReference reference_visible`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `putBefore` operation triggers the execution of the "PutBefore" elementary action with the attached `Visible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

The `reference_visible` parameter corresponds to the "reference_visible" parameter of the "PutBefore" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.17.6 `putBehind` operation

Synopsis:

Interface: `Visible`

Operation: `putBehind`

Result: `void`

In: `GenericObjectReference reference_visible`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `putBehind` operation triggers the execution of the "PutBehind" elementary action with the attached `Visible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.18.4.

The `reference_visible` parameter corresponds to the "reference_visible" parameter of the "PutBehind" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.17.7 IDL description

```
interface Visible: Presentable
{
    void
        setPosition(
            in  GenericInteger  new_x_position,
            in  GenericInteger  new_y_position)
    raises(InvalidTarget, InvalidParameter);

    void
        setBoxSize(
            in  GenericInteger  x_box_size,
            in  GenericInteger  y_box_size)
    raises(InvalidTarget, InvalidParameter);

    void
        bringToFront()
    raises(InvalidTarget);

    void
        sendToBack()
    raises(InvalidTarget);

    void
        putBefore(
            in  GenericObjectReference  reference_visible)
    raises(InvalidTarget, InvalidParameter);

    void
        putBehind(
            in  GenericObjectReference  reference_visible)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.18 `Bitmap` interface

The `Bitmap` interface inherits from the `Visible` interface.

The following subclauses define the operations provided by the `Bitmap` interface.

8.1.18.1 `setBitmapPaletteRef` operation

Synopsis:

Interface: `Bitmap`

Operation: `setBitmapPaletteRef`

Result: `void`

In: `GenericObjectReference  new_bitmap_palette`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setBitmapPaletteRef` operation triggers the execution of the "SetBitmapPaletteRef" elementary action with the attached `Bitmap` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.19.4.

The `new_bitmap_palette` parameter corresponds to the "new_bitmap_palette" parameter of the "SetBitmapPaletteRef" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.18.2 `scaleBitmap` operation

Synopsis:

Interface: `Bitmap`

Operation: `scaleBitmap`

Result: `void`

In: `GenericInteger` `x_scale`

In: `GenericInteger` `y_scale`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `scaleBitmap` operation triggers the execution of the "ScaleBitmap" elementary action with the attached `Bitmap` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.19.4.

The `x_scale` parameter corresponds to the "x_scale" parameter of the "ScaleBitmap" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `y_scale` parameter corresponds to the "y_scale" parameter of the "ScaleBitmap" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.18.3 `setTransparency` operation

Synopsis:

Interface: `Bitmap`

Operation: `setTransparency`

Result: `void`

In: `GenericInteger` `new_transparency`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setTransparency` operation triggers the execution of the "SetTransparency" elementary action with the attached `Bitmap` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.19.4.

The `new_transparency` parameter corresponds to the "new_transparency" parameter of the "SetTransparency" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.18.4 IDL description

```
interface Bitmap: Visible
{
    void
        setBitmapPaletteRef(
            in  GenericObjectReference  new_bitmap_palette)
    raises(InvalidTarget, InvalidParameter);

    void
        scaleBitmap(
            in  GenericInteger  x_scale,
            in  GenericInteger  y_scale)
    raises(InvalidTarget, InvalidParameter);

    void
        setTransparency(
            in  GenericInteger  new_transparency)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.19 `LineArt` interface

The `LineArt` interface inherits from the `Visible` interface.

The following subclauses define the operations provided by the `LineArt` interface.

8.1.19.1 `setLineWidth` operation

Synopsis:

Interface: `LineArt`

Operation: `setLineWidth`

Result: `void`

In: `long line_width`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setLineWidth` operation triggers the execution of the "SetLineWidth" elementary action with the attached `LineArt` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.20.4.

The `line_width` parameter corresponds to the "line_width" parameter of the "SetLineWidth" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.19.2 `setLineColour` operation

Synopsis:

Interface: `LineArt`

Operation: `setLineColour`

Result: `void`

In: `NewColour new_line_colour`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setLineColour` operation triggers the execution of the "SetLineColour" elementary action with the attached `LineArt` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.20.4.

The `new_line_colour` parameter corresponds to the "new_line_colour" parameter of the "SetLineColour" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.19.3 `setFillColour` operation

Synopsis:

Interface: `LineArt`

Operation: `setFillColour`

Result: `void`

In: `NewColour new_fill_colour`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setFillColour` operation triggers the execution of the "SetFillColour" elementary action with the attached `LineArt` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.20.4.

The `new_fill_colour` parameter corresponds to the "new_fill_colour" parameter of the "SetFillColour" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.19.4 `setLineartPalette` operation

Synopsis:

Interface: `LineArt`

Operation: `setLineartPalette`

Result: `void`

In: `GenericObjectReference new_lineart_palette`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setLineartPalette` operation triggers the execution of the "SetLineartPalette" elementary action with the attached `LineArt` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.20.4.

The `new_lineart_palette` parameter corresponds to the "new_lineart_palette" parameter of the "SetLineartPalette" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.19.5 IDL description

```
interface LineArt: Visible
{
    void
        setLineWidth(
            in  long    line_width)
    raises(InvalidTarget, InvalidParameter);

    void
        setLineColour(
            in  NewColour   new_line_colour)
    raises(InvalidTarget, InvalidParameter);

    void
        setFillColour(
            in  NewColour   new_fill_colour)
    raises(InvalidTarget, InvalidParameter);

    void
        setLineartPalette(
            in  GenericObjectReference  new_lineart_palette)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.20 `Rectangle` interface

The `Rectangle` interface inherits from the `LineArt` interface.

The `Rectangle` interface does not define any specific operation.

8.1.20.1 IDL description

```
interface Rectangle: LineArt
{};
```

8.1.21 `Text` interface

The `Text` interface inherits from the `Visible` interface.

The following subclauses define the operations provided by the `Text` interface.

8.1.21.1 `getTextData` operation

Synopsis:

Interface: `Text`

Operation: `getTextData`

Result: `TextData`

Exception: `InvalidTarget`

Description:

The `getTextData` operation triggers the execution of the "GetTextData" elementary action with the attached `Text` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.22.4.

8.1.21.2 `setFontRef` operation

Synopsis:

Interface: `Text`

Operation: `setFontRef`

Result: `void`

In: `NewFontRef new_font_ref`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setFontRef` operation triggers the execution of the "SetFontRef" elementary action with the attached `Text` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.22.4.

The `new_font_ref` parameter corresponds to the "new_font_ref" parameter of the "SetFontRef" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.21.3 `setTextPaletteRef` operation

Synopsis:

Interface: `Text`

Operation: `setTextPaletteRef`

Result: `void`

In: `GenericObjectReference new_text_palette`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setTextPaletteRef` operation triggers the execution of the "SetTextPaletteRef" elementary action with the attached `Text` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.22.4.

The new_text_palette parameter corresponds to the "new_text_palette" parameter of the "SetTextPaletteRef" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.21.4 IDL description

```
interface Text: Visible
{
    TextData
        getTextData()
    raises(InvalidTarget);

    void
        setFontRef(
            in  NewFontRef  new_font_ref)
    raises(InvalidTarget, InvalidParameter);

    void
        setTextPaletteRef(
            in  GenericObjectReference  new_text_palette)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.22 `Stream` interface

The Stream interface inherits from the Presentable interface.

The following subclauses define the operations provided by the Stream interface.

8.1.22.1 `setCounterTrigger` operation

Synopsis:

Interface: Stream

Operation: setCounterTrigger

Result: void

In: GenericInteger trigger_identifier

In: sequence<GenericInteger, 1> new_counter_value

Exception: InvalidTarget

Exception: InvalidParameter

Description:

The setCounterTrigger operation triggers the execution of the "SetCounterTrigger" elementary action with the attached Stream object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.23.4.

The trigger_identifier parameter corresponds to the "trigger_identifier" parameter of the "SetCounterTrigger" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The new_counter_value parameter corresponds to the "new_counter_value" parameter of the "SetCounterTrigger" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.22.2 `setSpeed` operation

Synopsis:

Interface: `Stream`

Operation: `setSpeed`

Result: `void`

In: `Rational new_speed`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setSpeed` operation triggers the execution of the "SetSpeed" elementary action with the attached `Stream` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.23.4.

The `new_speed` parameter corresponds to the "new_speed" parameter of the "SetSpeed" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.22.3 `setCounterPosition` operation

Synopsis:

Interface: `Stream`

Operation: `setCounterPosition`

Result: `void`

In: `GenericInteger new_counter_position`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setCounterPosition` operation triggers the execution of the "SetCounterPosition" elementary action with the attached `Stream` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.23.4.

The `new_counter_position` parameter corresponds to the "new_counter_position" parameter of the "SetCounterPosition" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.22.4 IDL description

```
interface Stream: Presentable
{
    void
        setCounterTrigger(
            in  GenericInteger  trigger_identifier,
            in  sequence<GenericInteger, 1> new_counter_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setSpeed(
            in  Rational    new_speed)
    raises(InvalidTarget, InvalidParameter);

    void
        setCounterPosition(
            in  GenericInteger  new_counter_position)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.23 `Audio` interface

The `Audio` interface inherits from the `Presentable` interface.

The following subclauses define the operations provided by the `Audio` interface.

8.1.23.1 `setVolume` operation

Synopsis:

Interface: `Audio`

Operation: `setVolume`

Result: `void`

In: `GenericInteger new_volume`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setVolume` operation triggers the execution of the "SetVolume" elementary action with the attached `Audio` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.24.4.

The `new_volume` parameter corresponds to the "new_volume" parameter of the "SetVolume" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.23.2 IDL description

```
interface Audio: Presentable
{
    void
        setVolume(
            in  GenericInteger  new_volume)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.24 `Video` interface

The `Video` interface inherits from the `Visible` interface.

The following subclauses define the operations provided by the `Video` interface.

8.1.24.1 `scaleVideo` operation

Synopsis:

Interface: `Video`

Operation: `scaleVideo`

Result: `void`

In: `GenericInteger x_scale`

In: `GenericInteger y_scale`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `scaleVideo` operation triggers the execution of the "ScaleVideo" elementary action with the attached `Video` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.25.4.

The `x_scale` parameter corresponds to the "x_scale" parameter of the "ScaleVideo" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

The `y_scale` parameter corresponds to the "y_scale" parameter of the "ScaleVideo" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.24.2 IDL description

```
interface Video: Visible
{
    void
        scaleVideo(
            in  GenericInteger  x_scale,
            in  GenericInteger  y_scale)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.25 `RTGraphics` interface

The `RTGraphics` interface inherits from the `Visible` interface.

The `RTGraphics` interface does not define any specific operation.

8.1.25.1 IDL description

```
interface RTGraphics: Visible
{};
```

8.1.26 `Interactible` interface

The following subclauses define the operations provided by the `Interactible` interface.

8.1.26.1 `setInteractionStatus` operation

Synopsis:

Interface: `Interactible`

Operation: `setInteractionStatus`

Result: `void`

In: `GenericBoolean new_interaction_status`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setInteractionStatus` operation triggers the execution of the "SetInteractionStatus" elementary action with the attached `Interactible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.27.4.

The `new_interaction_status` parameter corresponds to the "new_interaction_status" parameter of the "SetInteractionStatus" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.26.2 `setHighlightStatus` operation

Synopsis:

Interface: `Interactible`

Operation: `setHighlightStatus`

Result: `void`

In: `GenericBoolean new_highlight_status`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setHighlightStatus` operation triggers the execution of the "SetHighlightStatus" elementary action with the attached `Interactible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.27.4.

The `new_highlight_status` parameter corresponds to the "new_highlight_status" parameter of the "SetHighlightStatus" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.26.3 `setInteractiblePalette` operation

Synopsis:

Interface: `Interactible`

Operation: `setInteractiblePalette`

Result: `void`

In: `GenericObjectReference new_interactible_palette`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setInteractiblePalette` operation triggers the execution of the "SetInteractiblePalette" elementary action with the attached `Interactible` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.27.4.

The `new_interactible_palette` parameter corresponds to the "new_interactible_palette" parameter of the "SetInteractiblePalette" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.26.4 IDL description

```
interface Interactible
{
    void
        setInteractionStatus(
            in  GenericBoolean  new_interaction_status)
    raises(InvalidTarget, InvalidParameter);

    void
        setHighlightStatus(
            in  GenericBoolean  new_highlight_status)
    raises(InvalidTarget, InvalidParameter);

    void
        setInteractiblePalette(
            in  GenericObjectReference  new_interactible_palette)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.27 `Slider` interface

The `Slider` interface inherits from the `Visible` and the `Interactible` interfaces.

The following subclauses define the operations provided by the `Slider` interface.

8.1.27.1 `step` operation

Synopsis:

Interface: `Slider`

Operation: `step`

Result: `void`

In: `GenericInteger nb_of_steps`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `step` operation triggers the execution of the "Step" elementary action with the attached `Slider` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.28.4.

The `nb_of_steps` parameter corresponds to the "nb_of_steps" parameter of the "Step" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.27.2 `setSliderValue` operation

Synopsis:

Interface: `Slider`

Operation: `setSliderValue`

Result: `void`

In: `GenericInteger new_slider_value`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setSliderValue` operation triggers the execution of the "SetSliderValue" elementary action with the attached `Slider` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.28.4.

The `new_slider_value` parameter corresponds to the "new_slider_value" parameter of the "SetSliderValue" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.27.3 `setPortion` operation

Synopsis:

Interface: `Slider`

Operation: `setPortion`

Result: `void`

In: `GenericInteger new_portion`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setPortion` operation triggers the execution of the "SetPortion" elementary action with the attached `Slider` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.28.4.

The `new_portion` parameter corresponds to the "new_portion" parameter of the "SetPortion" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.27.4 `getSliderValue` operation

Synopsis:

Interface: `Slider`

Operation: `getSliderValue`

Result: `long`

Exception: `InvalidTarget`

Description:

The `getSliderValue` operation triggers the execution of the "GetSliderValue" elementary action with the attached `Slider` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.28.4.

8.1.27.5 IDL description

```
interface Slider: Visible, Interactible
{
    void
        step(
            in  GenericInteger  nb_of_steps)
    raises(InvalidTarget, InvalidParameter);

    void
        setSliderValue(
            in  GenericInteger  new_slider_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setPortion(
            in  GenericInteger  new_portion)
    raises(InvalidTarget, InvalidParameter);

    long
        getSliderValue()
    raises(InvalidTarget);
};
```

8.1.28 `EntryField` interface

The `EntryField` interface inherits from the `Text` and the `Interactible` interfaces.

The following subclauses define the operations provided by the `EntryField` interface.

8.1.28.1 `setOverwriteMode` operation

Synopsis:

Interface: `EntryField`

Operation: `setOverwriteMode`

Result: `void`

In: `GenericBoolean new_overwrite_mode`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setOverwriteMode` operation triggers the execution of the "SetOverwriteMode" elementary action with the attached `EntryField` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.29.4.

The `new_overwrite_mode` parameter corresponds to the "new_overwrite_mode" parameter of the "SetOverwriteMode" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.28.2 `getEntryPoint` operation

Synopsis:

Interface: `EntryField`

Operation: `getEntryPoint`

Result: `long`

Exception: `InvalidTarget`

Description:

The `getEntryPoint` operation triggers the execution of the "GetEntryPoint" elementary action with the attached `EntryField` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.29.4.

8.1.28.3 `setEntryPoint` operation

Synopsis:

Interface: `EntryField`

Operation: `setEntryPoint`

Result: `void`

In: `GenericInteger new_entry_point`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setEntryPoint` operation triggers the execution of the "SetEntryPoint" elementary action with the attached `EntryField` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.29.4.

The `new_entry_point` parameter corresponds to the "new_entry_point" parameter of the "SetEntryPoint" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.28.4 IDL description

```
interface EntryField: Text, Interactible
{
    void
        setOverwriteMode(
            in  GenericBoolean  new_overwrite_mode)
    raises(InvalidTarget, InvalidParameter);

    long
        getEntryPoint()
    raises(InvalidTarget);

    void
        setEntryPoint(
            in  GenericInteger  new_entry_point)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.29 `HyperText` interface

The `HyperText` interface inherits from the `Text` and the `Interactible` interfaces.

The `HyperText` interface does not define any specific operation.

8.1.29.1 IDL description

```
interface HyperText: Text, Interactible
{};
```

8.1.30 `Button` interface

The `Button` interface inherits from the `Visible` and the `Interactible` interfaces.

The following subclauses define the operations provided by the `Button` interface.

8.1.30.1 `select` operation

Synopsis:

Interface: `Button`

Operation: `select`

Result: `void`

Exception: `InvalidTarget`

Description:

The `select` operation triggers the execution of the "Select" elementary action with the attached `Button` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.31.4.

8.1.30.2 `deselect` operation

Synopsis:

Interface: `Button`

Operation: `deselect`

Result: `void`

Exception: `InvalidTarget`

Description:

The `deselect` operation triggers the execution of the "Deselect" elementary action with the attached `Button` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.31.4.

8.1.30.3 IDL description

```
interface Button: Visible, Interactible
{
    void
        select()
    raises(InvalidTarget);

    void
        deselect()
    raises(InvalidTarget);
};
```

8.1.31 `Hotspot` interface

The `Hotspot` interface inherits from the `Button` interface.

The following subclauses define the operations provided by the `Hotspot` interface.

8.1.31.1 `select` operation

Synopsis:

Interface: `Hotspot`

Operation: `select`

Result: `void`

Exception: `InvalidTarget`

Description:

The `select` operation triggers the execution of the "Select" elementary action with the attached `Hotspot` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.32.4.

Note that this operation does not appear in the interface IDL definition as it overrides the semantics of the inherited operation provided by the `Button` interface.

8.1.31.2 IDL description

```
interface Hotspot: Button
{
    // the ‘select’ operation is defined in the ‘Button’ interface
};
```

8.1.32 `PushButton` interface

The `PushButton` interface inherits from the `Button` interface.

The following subclauses define the operations provided by the `PushButton` interface.

8.1.32.1 `select` operation

Synopsis:

Interface: `PushButton`

Operation: `select`

Result: `void`

Exception: `InvalidTarget`

Description:

The `select` operation triggers the execution of the "Select" elementary action with the attached `PushButton` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.33.4.

Note that this operation does not appear in the interface IDL definition as it overrides the semantics of the inherited operation provided by the `Button` interface.

8.1.32.2 `setLabel` operation

Synopsis:

Interface: `PushButton`

Operation: `setLabel`

Result: `void`

In: `GenericOctetString new_label`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setLabel` operation triggers the execution of the "SetLabel" elementary action with the attached `PushButton` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.33.4.

The `new_label` parameter corresponds to the "new_label" parameter of the "SetLabel" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.32.3 IDL description

```
interface PushButton: Button
{
    // the ‘select’ operation is defined in the ‘Button’ interface

    void
        setLabel(
            in  GenericOctetString  new_label)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.33 `SwitchButton` interface

The `SwitchButton` interface inherits from the `Button` interface.

The following subclauses define the operations provided by the `SwitchButton` interface.

8.1.33.1 `getSelectionStatus` operation

Synopsis:

Interface: `SwitchButton`

Operation: `getSelectionStatus`

Result: `boolean`

Exception: `InvalidTarget`

Description:

The `getSelectionStatus` operation triggers the execution of the "GetSelectionStatus" elementary action with the attached `SwitchButton` object as its target.

The effect of the action on its target and the computation of its result are defined by ISO/IEC DIS 13522-5 [1], subclause 2.34.4.

8.1.33.2 `deselect` operation

Synopsis:

Interface: `SwitchButton`

Operation: `deselect`

Result: `void`

Exception: `InvalidTarget`

Description:

The `deselect` operation triggers the execution of the "Deselect" elementary action with the attached `SwitchButton` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.34.4.

Note that this operation does not appear in the interface IDL definition as it overrides the semantics of the inherited operation provided by the `Button` interface.

8.1.33.3 `toggle` operation

Synopsis:

Interface: `SwitchButton`

Operation: `toggle`

Result: `void`

Exception: `InvalidTarget`

Description:

The `toggle` operation triggers the execution of the "Toggle" elementary action with the attached `SwitchButton` object as its target.

The effect of the action on its target is defined by ISO/IEC DIS 13522-5 [1], subclause 2.34.4.

8.1.33.4 `setLabel` operation

Synopsis:

Interface: `SwitchButton`

Operation: `setLabel`

Result: `void`

In: `GenericOctetString new_label`

Exception: `InvalidTarget`

Exception: `InvalidParameter`

Description:

The `setLabel` operation triggers the execution of the "SetLabel" elementary action with the attached `SwitchButton` object as its target.

The effect of the action on its target and the semantics of its parameters are defined by ISO/IEC DIS 13522-5 [1], subclause 2.34.4.

The `new_label` parameter corresponds to the "new_label" parameter of the "SetLabel" action in the ASN.1 definition (see annex A of ISO/IEC DIS 13522-5 [1]).

8.1.33.5 IDL description

```
interface SwitchButton: Button
{
    boolean
        getSelectionStatus()
    raises(InvalidTarget);

    // the ‘deselect’ operation is defined in the ‘Button’ interface

    void
        toggle()
    raises(InvalidTarget);

    void
        setLabel(
            in  GenericOctetString  new_label)
    raises(InvalidTarget, InvalidParameter);
};
```

8.1.34 DL definition

```
module MHEG_5 {

// Corresponding MHEG datatype: EventType
//=============================================================
enum EventType { IS_AVAILABLE, IS_DELETED, IS_RUNNING, IS_STOPPED, USER_INPUT, TIMER_FIRED,
ASYNCH_STOPPED, IS_MODIFIED, TOKEN_MOVED_FROM, TOKEN_MOVED_TO, STREAM_EVENT, STREAM_PLAYING,
STREAM_STOPPED, COUNTER_TRIGGER, HIGHLIGHT_ON, HIGHLIGHT_OFF, CURSOR_ENTER, CURSOR_LEAVE,
NEW_CHAR, ANCHOR_FIRED, IS_SELECTED, IS_DESELECTED };

// Corresponding MHEG datatype: ScrollType
//=============================================================
enum ScrollType { S_T_UP, S_T_DOWN, S_T_ABSOLUTE };

// Corresponding MHEG datatype: ObjectReference
//=============================================================
struct ObjectReference {
    sequence<octet> group_identifier;
    long    object_number;
};

// Corresponding MHEG datatype: ContentReference
//=============================================================
struct ContentReference {
    sequence<octet> public_identifier;
    sequence<octet> system_identifier;
};

// Corresponding MHEG datatype: GenericObjectReference
//=============================================================
enum GenericObjectReferenceTag { DIRECT_REFERENCE_TAG, INDIRECT_REFERENCE_TAG };
union GenericObjectReference
switch (GenericObjectReferenceTag) {
    case DIRECT_REFERENCE_TAG:
        ObjectReference direct_reference;
    case INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericContentReference
//=============================================================
enum GenericContentReferenceTag { G_C_R_CONTENT_REFERENCE_TAG, G_C_R_INDIRECT_REFERENCE_TAG };
union GenericContentReference
switch (GenericContentReferenceTag) {
    case G_C_R_CONTENT_REFERENCE_TAG:
        ContentReference    content_reference;
    case G_C_R_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericInteger
//=============================================================
enum GenericIntegerTag { G_I_INTEGER_VALUE_TAG, G_I_INDIRECT_REFERENCE_TAG };
union GenericInteger
switch (GenericIntegerTag) {
    case G_I_INTEGER_VALUE_TAG:
        long    integer_value;
    case G_I_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericBoolean
//=============================================================
enum GenericBooleanTag { G_B_BOOLEAN_VALUE_TAG, G_B_INDIRECT_REFERENCE_TAG };
union GenericBoolean
switch (GenericBooleanTag) {
    case G_B_BOOLEAN_VALUE_TAG:
        boolean boolean_value;
    case G_B_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericOctetString
//=============================================================
enum GenericOctetStringTag { G_O_S_OCTETSTRING_VALUE_TAG, G_O_S_INDIRECT_REFERENCE_TAG };
union GenericOctetString
switch (GenericOctetStringTag) {
    case G_O_S_OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
    case G_O_S_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
```

```
};

// Corresponding MHEG datatype: NewVisibleContent
//=============================================================
enum NewVisibleContentTag { N_V_C_NEW_CONTENT_REFERENCE_TAG, N_V_C_NULL_TAG };
union NewVisibleContent
switch (NewVisibleContentTag) {
    case N_V_C_NEW_CONTENT_REFERENCE_TAG:
        GenericContentReference new_content_reference;
};

// Corresponding MHEG datatype: NewVariableContent
//=============================================================
enum NewVariableContentTag { NEW_VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG,
NEW_VARIABLE_CONTENT_INTEGER_VALUE_TAG, NEW_VARIABLE_CONTENT_BOOLEAN_VALUE_TAG,
NEW_VARIABLE_CONTENT_OBJECT_REFERENCE_TAG, NEW_VARIABLE_CONTENT_CONTENT_REFERENCE_TAG,
NEW_VARIABLE_CONTENT_NULL_TAG };
union NewVariableContent
switch (NewVariableContentTag) {
    case NEW_VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG:
        GenericOctetString  octetstring_value;
    case NEW_VARIABLE_CONTENT_INTEGER_VALUE_TAG:
        GenericInteger  integer_value;
    case NEW_VARIABLE_CONTENT_BOOLEAN_VALUE_TAG:
        GenericBoolean  boolean_value;
    case NEW_VARIABLE_CONTENT_OBJECT_REFERENCE_TAG:
        GenericObjectReference  object_reference;
    case NEW_VARIABLE_CONTENT_CONTENT_REFERENCE_TAG:
        GenericContentReference content_reference;
};

// Corresponding MHEG datatype: EmulatedEventData
//=============================================================
enum EmulatedEventDataTag { GENERIC_BOOLEAN_TAG, GENERIC_INTEGER_TAG, GENERIC_OCTET_STRING_TAG };
union EmulatedEventData
switch (EmulatedEventDataTag) {
    case GENERIC_BOOLEAN_TAG:
        GenericBoolean  generic_boolean;
    case GENERIC_INTEGER_TAG:
        GenericInteger  generic_integer;
    case GENERIC_OCTET_STRING_TAG:
        GenericOctetString  generic_octet_string;
};

// Corresponding MHEG datatype: NewContent
//=============================================================
enum NewContentTag { NEW_INCLUDED_CONTENT_TAG, NEW_REFERENCED_CONTENT_TAG };
union NewContent
switch (NewContentTag) {
    case NEW_INCLUDED_CONTENT_TAG:
        GenericOctetString  new_included_content;
    case NEW_REFERENCED_CONTENT_TAG:
        GenericContentReference new_referenced_content;
};

// Corresponding MHEG datatype: NewColour
//=============================================================
enum NewColourTag { N_C_COLOUR_INDEX_TAG, N_C_ABSOLUTE_COLOUR_TAG };
union NewColour
switch (NewColourTag) {
    case N_C_COLOUR_INDEX_TAG:
        GenericInteger  colour_index;
    case N_C_ABSOLUTE_COLOUR_TAG:
        GenericOctetString  absolute_colour;
};

// Corresponding MHEG datatype: NewFontRef
//=============================================================
enum NewFontRefTag { N_F_R_FONT_NAME_TAG, N_F_R_FONT_REFERENCE_TAG };
union NewFontRef
switch (NewFontRefTag) {
    case N_F_R_FONT_NAME_TAG:
        GenericOctetString  font_name;
    case N_F_R_FONT_REFERENCE_TAG:
        ObjectReference font_reference;
};

// Corresponding MHEG datatype: Rational
//=============================================================
struct Rational {
    GenericInteger  nominator;
    sequence<GenericInteger,1>  denominator;
};
```

```
// Corresponding MHEG datatype: NewVariableValue
//=================================================================
enum NewVariableValueTag { N_V_V_GENERIC_INTEGER_TAG, N_V_V_GENERIC_BOOLEAN_TAG,
N_V_V_GENERIC_OCTET_STRING_TAG, N_V_V_GENERIC_OBJECT_REFERENCE_TAG,
N_V_V_GENERIC_CONTENT_REFERENCE_TAG };
union NewVariableValue
switch (NewVariableValueTag) {
    case N_V_V_GENERIC_INTEGER_TAG:
        GenericInteger  generic_integer;
    case N_V_V_GENERIC_BOOLEAN_TAG:
        GenericBoolean  generic_boolean;
    case N_V_V_GENERIC_OCTET_STRING_TAG:
        GenericOctetString  generic_octet_string;
    case N_V_V_GENERIC_OBJECT_REFERENCE_TAG:
        GenericObjectReference  generic_object_reference;
    case N_V_V_GENERIC_CONTENT_REFERENCE_TAG:
        GenericContentReference generic_content_reference;
};

// Corresponding MHEG datatype: NewItemData
//=================================================================
struct NewItemData {
    sequence<NewVisibleContent> new_visibles_content;
    sequence<NewVariableContent>    new_variables_content;
};

// Interface : Scene, Operation : getCursorPosition
// Corresponding MHEG-5 datatype : none
//=================================================================================

struct CursorPosition {
        long    x;
        long    y;
};

// Interface : Procedure, Operation : runSynchronous
// Interface : Procedure, Operation : runAsynchronous
// Corresponding MHEG-5 datatype : none
// (derives from the InParameters MHEG-5 datatype)
//=================================================================================

struct Parameters {
        sequence<GenericBoolean>                booleans;
        sequence<GenericInteger>                integers;
        sequence<GenericOctetString>            octetstrings;
        sequence<GenericObjectReference>        object_references;
        sequence<GenericContentReference>       content_references;
};

// Interface : Variable, Operation : getVariable
// Corresponding MHEG datatype: none
//=================================================================
enum VariableValueTag { V_V_INTEGER_TAG, V_V_BOOLEAN_TAG, V_V_OCTET_STRING_TAG,
V_V_OBJECT_REFERENCE_TAG, V_V_CONTENT_REFERENCE_TAG };
union VariableValue
switch (VariableValueTag) {
        case V_V_INTEGER_TAG:
                long                    integer_value;
        case V_V_BOOLEAN_TAG:
                boolean                 boolean_value;
        case V_V_OCTET_STRING_TAG:
                sequence<octet>         octetstring_value;
        case V_V_OBJECT_REFERENCE_TAG:
                ObjectReference         object_reference;
        case V_V_CONTENT_REFERENCE_TAG:
                ContentReference        content_reference;
};

// Interface : Text, Operation : getTextData
// Corresponding MHEG-5 datatype : none
//=================================================================================

enum TextDataTag { T_D_OCTET_STRING_TAG, T_D_CONTENT_REFERENCE_TAG, T_D_NULL_TAG };
union TextData
switch (TextDataTag) {
        case    T_D_OCTET_STRING_TAG:       sequence<octet>         octetstring_value;
        case    T_D_CONTENT_REFERENCE_TAG:  ContentReference        content_reference;
};

// Exceptions
//=================================================================================
```

```
exception InvalidTarget {};

enum CompletionStatus { YES, NO };

exception InvalidParameter {
        CompletionStatus        completion_status;
        unsigned short          rank;
};

interface Root
{
    boolean
        getAvailabilityStatus()
    raises(InvalidTarget);

    boolean
        getRunningStatus()
    raises(InvalidTarget);
};

interface Group: Root
{
    void
        setCachePriority(
            in  GenericInteger  new_cache_priority)
    raises(InvalidTarget, InvalidParameter);
};

interface Application: Group
{
    boolean
        storePersistent(
            in  sequence<ObjectReference>   in_variables,
            in  GenericOctetString  out_file_name)
    raises(InvalidTarget, InvalidParameter);

    boolean
        readPersistent(
            in  sequence<ObjectReference>   out_variables,
            in  GenericOctetString  in_file_name)
    raises(InvalidTarget, InvalidParameter);

    void
        launch()
    raises(InvalidTarget);

    void
        spawn()
    raises(InvalidTarget);

    void
        quit()
    raises(InvalidTarget);

    void
        lockScreen()
    raises(InvalidTarget);

    void
        unlockScreen()
    raises(InvalidTarget);

    boolean
        openConnection(
            in  GenericOctetString  protocol,
            in  GenericOctetString  address,
            in  GenericInteger  connection_tag)
    raises(InvalidTarget, InvalidParameter);

    void
        closeConnection(
            in  GenericInteger  connection_tag)
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getEngineSupport()
    raises(InvalidTarget);
};

interface Scene: Group
{
    void
        transitionTo(
```

```
            in  sequence<GenericInteger, 1> connection_tag,
            in  GenericInteger  transition_effect)
    raises(InvalidTarget, InvalidParameter);

    void
        setTimer(
            in  GenericInteger  timer_id,
            in  sequence<GenericInteger, 1> timer_value)
    raises(InvalidTarget, InvalidParameter);

    void
        sendEvent(
            in  ObjectReference event_source,
            in  EventType   event_type,
            in  sequence<EmulatedEventData, 1>  emulated_event_data)
    raises(InvalidTarget, InvalidParameter);

    void
        setCursorShape(
            in  sequence<GenericObjectReference, 1> new_cursor_shape)
    raises(InvalidTarget, InvalidParameter);

    void
        setCursorPosition(
            in  GenericInteger  x_cursor,
            in  GenericInteger  y_cursor)
    raises(InvalidTarget, InvalidParameter);

    CursorPosition
        getCursorPosition()
    raises(InvalidTarget);
};

interface Ingredient: Root
{
    void
        setData(
            in  NewContent  new_content)
    raises(InvalidTarget, InvalidParameter);

    void
        preload()
    raises(InvalidTarget);

    void
        unload()
    raises(InvalidTarget);
};

interface Link: Ingredient
{
    void
        activate()
    raises(InvalidTarget);

    void
        deactivate()
    raises(InvalidTarget);
};

interface Procedure: Ingredient
{
    boolean
        runSynchronous(
            in  Parameters  in_parameters)
    raises(InvalidTarget, InvalidParameter);

    boolean
        runAsynchronous(
            in  Parameters  in_parameters)
    raises(InvalidTarget, InvalidParameter);

    void
        stop()
    raises(InvalidTarget);
};

interface Palette: Ingredient
{
};

interface Font: Ingredient
{
```

```
};

interface CursorShape: Ingredient
{
};

interface Variable: Ingredient
{
    void
        setVariable(
            in  NewVariableValue    new_variable_value)
    raises(InvalidTarget, InvalidParameter);

    VariableValue
        getVariable()
    raises(InvalidTarget);
};

interface Presentable: Ingredient
{
    // the ‘setData’ operation is defined in the ‘Ingredient’ interface

    void
        run()
    raises(InvalidTarget);

    void
        stop()
    raises(InvalidTarget);
};

interface TokenManager
{
    void
        move(
            in  GenericInteger  movement_identifier)
    raises(InvalidTarget, InvalidParameter);

    void
        moveTo(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);
};

interface TokenGroup: Presentable, TokenManager
{
    void
        callActionSlot(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);
};

interface TemplateGroup: Presentable, TokenManager
{
    void
        selectItem()
    raises(InvalidTarget);

    void
        deselectItem()
    raises(InvalidTarget);

    void
        toggleItem()
    raises(InvalidTarget);

    void
        callTemplateAction(
            in  GenericInteger  index)
    raises(InvalidTarget, InvalidParameter);

    void
        setState(
            in  GenericOctetString  new_state)
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getState()
    raises(InvalidTarget);

    void
        setItemContent(
            in  GenericOctetString  new_items_content)
```

```
    raises(InvalidTarget, InvalidParameter);

    sequence<octet>
        getItemContent()
    raises(InvalidTarget);
};

interface List: TemplateGroup
{
    void
        plug(
            in  NewItemData item_data,
            in  sequence<GenericInteger, 1> index)
    raises(InvalidTarget, InvalidParameter);

    void
        unplug(
            in  sequence<GenericInteger, 1> index)
    raises(InvalidTarget, InvalidParameter);

    void
        scroll(
            in  ScrollType  scroll_type,
            in  GenericInteger  scroll_number)
    raises(InvalidTarget, InvalidParameter);
};

interface Visible: Presentable
{
    void
        setPosition(
            in  GenericInteger  new_x_position,
            in  GenericInteger  new_y_position)
    raises(InvalidTarget, InvalidParameter);

    void
        setBoxSize(
            in  GenericInteger  x_box_size,
            in  GenericInteger  y_box_size)
    raises(InvalidTarget, InvalidParameter);

    void
        bringToFront()
    raises(InvalidTarget);

    void
        sendToBack()
    raises(InvalidTarget);

    void
        putBefore(
            in  GenericObjectReference  reference_visible)
    raises(InvalidTarget, InvalidParameter);

    void
        putBehind(
            in  GenericObjectReference  reference_visible)
    raises(InvalidTarget, InvalidParameter);
};

interface Bitmap: Visible
{
    void
        setBitmapPaletteRef(
            in  GenericObjectReference  new_bitmap_palette)
    raises(InvalidTarget, InvalidParameter);

    void
        scaleBitmap(
            in  GenericInteger  x_scale,
            in  GenericInteger  y_scale)
    raises(InvalidTarget, InvalidParameter);

    void
        setTransparency(
            in  GenericInteger  new_transparency)
    raises(InvalidTarget, InvalidParameter);
};

interface LineArt: Visible
{
    void
        setLineWidth(
```

```
            in  long    line_width)
    raises(InvalidTarget, InvalidParameter);

    void
        setLineColour(
            in  NewColour   new_line_colour)
    raises(InvalidTarget, InvalidParameter);

    void
        setFillColour(
            in  NewColour   new_fill_colour)
    raises(InvalidTarget, InvalidParameter);

    void
        setLineartPalette(
            in  GenericObjectReference  new_lineart_palette)
    raises(InvalidTarget, InvalidParameter);
};

interface Rectangle: LineArt
{
};

interface Text: Visible
{
    TextData
        getTextData()
    raises(InvalidTarget);

    void
        setFontRef(
            in  NewFontRef  new_font_ref)
    raises(InvalidTarget, InvalidParameter);

    void
        setTextPaletteRef(
            in  GenericObjectReference  new_text_palette)
    raises(InvalidTarget, InvalidParameter);
};

interface Stream: Presentable
{
    void
        setCounterTrigger(
            in  GenericInteger  trigger_identifier,
            in  sequence<GenericInteger, 1> new_counter_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setSpeed(
            in  Rational    new_speed)
    raises(InvalidTarget, InvalidParameter);

    void
        setCounterPosition(
            in  GenericInteger  new_counter_position)
    raises(InvalidTarget, InvalidParameter);
};

interface Audio: Presentable
{
    void
        setVolume(
            in  GenericInteger  new_volume)
    raises(InvalidTarget, InvalidParameter);
};

interface Video: Visible
{
    void
        scaleVideo(
            in  GenericInteger  x_scale,
            in  GenericInteger  y_scale)
    raises(InvalidTarget, InvalidParameter);
};

interface RTGraphics: Visible
{
};

interface Interactible
{
    void
```

```
        setInteractionStatus(
            in  GenericBoolean  new_interaction_status)
    raises(InvalidTarget, InvalidParameter);

    void
        setHighlightStatus(
            in  GenericBoolean  new_highlight_status)
    raises(InvalidTarget, InvalidParameter);

    void
        setInteractiblePalette(
            in  GenericObjectReference  new_interactible_palette)
    raises(InvalidTarget, InvalidParameter);
};

interface Slider: Visible, Interactible
{
    void
        step(
            in  GenericInteger  nb_of_steps)
    raises(InvalidTarget, InvalidParameter);

    void
        setSliderValue(
            in  GenericInteger  new_slider_value)
    raises(InvalidTarget, InvalidParameter);

    void
        setPortion(
            in  GenericInteger  new_portion)
    raises(InvalidTarget, InvalidParameter);

    long
        getSliderValue()
    raises(InvalidTarget);
};

interface EntryField: Text, Interactible
{
    void
        setOverwriteMode(
            in  GenericBoolean  new_overwrite_mode)
    raises(InvalidTarget, InvalidParameter);

    long
        getEntryPoint()
    raises(InvalidTarget);

    void
        setEntryPoint(
            in  GenericInteger  new_entry_point)
    raises(InvalidTarget, InvalidParameter);
};

interface HyperText: Text, Interactible
{
};

interface Button: Visible, Interactible
{
    void
        select()
    raises(InvalidTarget);

    void
        deselect()
    raises(InvalidTarget);
};

interface Hotspot: Button
{
    // the ‘select’ operation is defined in the ‘Button’ interface
};

interface PushButton: Button
{
    // the ‘select’ operation is defined in the ‘Button’ interface

    void
        setLabel(
            in  GenericOctetString  new_label)
    raises(InvalidTarget, InvalidParameter);
};
```

```
interface SwitchButton: Button
{
    boolean
        getSelectionStatus()
    raises(InvalidTarget);

    // the ‘deselect’ operation is defined in the ‘Button’ interface

    void
        toggle()
    raises(InvalidTarget);

    void
        setLabel(
            in  GenericOctetString  new_label)
    raises(InvalidTarget, InvalidParameter);
};

};
```

8.2 Access module

8.2.1 DL definition

```
module MHEG_5_Access    {

// Forward declarations of the interfaces
//=============================================================
interface RootClass;
interface GroupClass;
interface ApplicationClass;
interface SceneClass;
interface IngredientClass;
interface LinkClass;
interface ProcedureClass;
interface PaletteClass;
interface FontClass;
interface CursorShapeClass;
interface VariableClass;
interface PresentableClass;
interface TokenManagerClass;
interface TokenGroupClass;
interface TemplateGroupClass;
interface ListClass;
interface VisibleClass;
interface BitmapClass;
interface LineArtClass;
interface RectangleClass;
interface TextClass;
interface StreamClass;
interface AudioClass;
interface VideoClass;
interface RTGraphicsClass;
interface InteractibleClass;
interface SliderClass;
interface EntryFieldClass;
interface HypertextClass;
interface ButtonClass;
interface HotspotClass;
interface PushButtonClass;
interface SwitchButtonClass;
interface ActionClass;

// Corresponding MHEG datatype: OBJECT IDENTIFIER
//=============================================================
struct ObjectIdentifier {
    unsigned short  root;
    unsigned short  arc1;
    unsigned short  arc2;
    unsigned short  arc3;
    unsigned short  arc4;
};

// Corresponding MHEG datatype: Item
//=============================================================
enum ItemTag { PROCEDURE_CLASS_TAG, PALETTE_CLASS_TAG, FONT_CLASS_TAG, CURSOR_SHAPE_CLASS_TAG,
VARIABLE_CLASS_TAG, LINK_CLASS_TAG, STREAM_CLASS_TAG, AUDIO_CLASS_TAG, VIDEO_CLASS_TAG,
RTGRAPHICS_CLASS_TAG, BITMAP_CLASS_TAG, LINEART_CLASS_TAG, RECTANGLE_CLASS_TAG,
HOTSPOT_CLASS_TAG, SWITCH_BUTTON_CLASS_TAG, PUSH_BUTTON_CLASS_TAG, TEXT_CLASS_TAG,
ENTRYFIELD_CLASS_TAG, HYPERTEXT_CLASS_TAG, SLIDER_CLASS_TAG, TOKEN_GROUP_CLASS_TAG,
TEMPLATE_GROUP_CLASS_TAG, LIST_CLASS_TAG };
union Item
switch (ItemTag) {
    case PROCEDURE_CLASS_TAG:
        ProcedureClass  procedure_class;
    case PALETTE_CLASS_TAG:
        PaletteClass    palette_class;
    case FONT_CLASS_TAG:
        FontClass   font_class;
    case CURSOR_SHAPE_CLASS_TAG:
        CursorShapeClass    cursor_shape_class;
    case VARIABLE_CLASS_TAG:
        VariableClass   variable_class;
    case LINK_CLASS_TAG:
        LinkClass   link_class;
    case STREAM_CLASS_TAG:
        StreamClass stream_class;
    case AUDIO_CLASS_TAG:
        AudioClass  audio_class;
    case VIDEO_CLASS_TAG:
        VideoClass  video_class;
    case RTGRAPHICS_CLASS_TAG:
```

```
        RTGraphicsClass rtgraphics_class;
    case BITMAP_CLASS_TAG:
        BitmapClass bitmap_class;
    case LINEART_CLASS_TAG:
        LineArtClass    lineart_class;
    case RECTANGLE_CLASS_TAG:
        RectangleClass  rectangle_class;
    case HOTSPOT_CLASS_TAG:
        HotspotClass    hotspot_class;
    case SWITCH_BUTTON_CLASS_TAG:
        SwitchButtonClass   switch_button_class;
    case PUSH_BUTTON_CLASS_TAG:
        PushButtonClass push_button_class;
    case TEXT_CLASS_TAG:
        TextClass   text_class;
    case ENTRYFIELD_CLASS_TAG:
        EntryFieldClass entryfield_class;
    case HYPERTEXT_CLASS_TAG:
        HypertextClass  hypertext_class;
    case SLIDER_CLASS_TAG:
        SliderClass slider_class;
    case TOKEN_GROUP_CLASS_TAG:
        TokenGroupClass token_group_class;
    case TEMPLATE_GROUP_CLASS_TAG:
        TemplateGroupClass  template_group_class;
    case LIST_CLASS_TAG:
        ListClass   list_class;
};

// Corresponding MHEG datatype: SceneCoordinateSystem
//=============================================================
struct SceneCoordinateSystem {
    long    x_scene;
    long    y_scene;
};

// Corresponding MHEG datatype: AspectRatio
//=============================================================
struct AspectRatio {
    sequence<long,1>    width;
    sequence<long,1>    height;
};

// Corresponding MHEG datatype: NextScene
//=============================================================
struct NextScene {
    sequence<octet> scene_ref;
    long    scene_weight;
};

// Corresponding MHEG datatype: ContentHook
//=============================================================
enum ContentHookTag { MHEG_CONTENT_REGISTER_TAG, PROPRIETARY_CONTENT_REGISTER_TAG };
union ContentHook
switch (ContentHookTag) {
    case MHEG_CONTENT_REGISTER_TAG:
        long    mheg_content_register;
    case PROPRIETARY_CONTENT_REGISTER_TAG:
        long    proprietary_content_register;
};

// Corresponding MHEG datatype: EventType
//=============================================================
enum EventType { IS_AVAILABLE, IS_DELETED, IS_RUNNING, IS_STOPPED, USER_INPUT, TIMER_FIRED,
ASYNCH_STOPPED, IS_MODIFIED, TOKEN_MOVED_FROM, TOKEN_MOVED_TO, STREAM_EVENT, STREAM_PLAYING,
STREAM_STOPPED, COUNTER_TRIGGER, HIGHLIGHT_ON, HIGHLIGHT_OFF, CURSOR_ENTER, CURSOR_LEAVE,
NEW_CHAR, ANCHOR_FIRED, IS_SELECTED, IS_DESELECTED };

// Corresponding MHEG datatype: EventData
//=============================================================
enum EventDataTag { OCTETSTRING_VALUE_TAG, BOOLEAN_VALUE_TAG, INTEGER_VALUE_TAG };
union EventData
switch (EventDataTag) {
    case OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
    case BOOLEAN_VALUE_TAG:
        boolean boolean_value;
    case INTEGER_VALUE_TAG:
        long    integer_value;
};

// Corresponding MHEG datatype: ProcedureType
//=============================================================
```

```
enum ProcedureType { CUSTOM, REMOTE, SCRIPT };

// Corresponding MHEG datatype: Movement
//=============================================================
typedef sequence<long>  Movement;

// Corresponding MHEG datatype: Avisible
//=============================================================
enum AvisibleTag { A_TEXT_CLASS_TAG, A_VIDEO_CLASS_TAG, A_RTGRAPHICS_CLASS_TAG,
A_BITMAP_CLASS_TAG, A_LINEART_CLASS_TAG, A_RECTANGLE_CLASS_TAG, A_HOTSPOT_CLASS_TAG,
A_SWITCH_BUTTON_CLASS_TAG, A_PUSH_BUTTON_CLASS_TAG, A_SLIDER_CLASS_TAG, A_HYPERTEXT_CLASS_TAG,
A_ENTRYFIELD_CLASS_TAG };
union Avisible
switch (AvisibleTag) {
    case A_TEXT_CLASS_TAG:
        TextClass   text_class;
    case A_VIDEO_CLASS_TAG:
        VideoClass  video_class;
    case A_RTGRAPHICS_CLASS_TAG:
        RTGraphicsClass rtgraphics_class;
    case A_BITMAP_CLASS_TAG:
        BitmapClass bitmap_class;
    case A_LINEART_CLASS_TAG:
        LineArtClass    lineart_class;
    case A_RECTANGLE_CLASS_TAG:
        RectangleClass  rectangle_class;
    case A_HOTSPOT_CLASS_TAG:
        HotspotClass    hotspot_class;
    case A_SWITCH_BUTTON_CLASS_TAG:
        SwitchButtonClass   switch_button_class;
    case A_PUSH_BUTTON_CLASS_TAG:
        PushButtonClass push_button_class;
    case A_SLIDER_CLASS_TAG:
        SliderClass slider_class;
    case A_HYPERTEXT_CLASS_TAG:
        HypertextClass  hypertext_class;
    case A_ENTRYFIELD_CLASS_TAG:
        EntryFieldClass entryfield_class;
};

// Corresponding MHEG datatype: ActionSlot
//=============================================================
enum ActionSlotTag { ACTION_TAG, NULL_TAG };
union ActionSlot
switch (ActionSlotTag) {
    case ACTION_TAG:
        ActionClass action;
};

// Corresponding MHEG datatype: TemplateVisible
//=============================================================
struct TemplateVisible {
    Avisible    a_visible;
    boolean relative_to_scene;
};

// Corresponding MHEG datatype: TemplateAction
//=============================================================
enum TemplateActionTag { T_A_ACTION_TAG, T_A_NULL_TAG };
union TemplateAction
switch (TemplateActionTag) {
    case T_A_ACTION_TAG:
        ActionClass action;
};

// Corresponding MHEG datatype: CellPosition
//=============================================================
struct CellPosition {
    long    x_cell;
    long    y_cell;
};

// Corresponding MHEG datatype: OriginalBoxSize
//=============================================================
struct OriginalBoxSize {
    long    x_length;
    long    y_length;
};

// Corresponding MHEG datatype: OriginalPosition
//=============================================================
struct OriginalPosition {
    sequence<long,1>    x_position;
```

```
    sequence<long,1>    y_position;
};

// Corresponding MHEG datatype: DropOutColour
//=============================================================
enum DropOutColourTag { D_O_C_INTEGER_VALUE_TAG, D_O_C_OCTETSTRING_VALUE_TAG };
union DropOutColour
switch (DropOutColourTag) {
    case D_O_C_INTEGER_VALUE_TAG:
        long    integer_value;
    case D_O_C_OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
};

// Corresponding MHEG datatype: LineStyle
//=============================================================
enum LineStyle { SOLID, DASHED, DOTTED };

// Corresponding MHEG datatype: HorizontalJustification
//=============================================================
enum HorizontalJustification { START, END, CENTRE, JUSTIFIED };

// Corresponding MHEG datatype: VerticalJustification
//=============================================================
enum VerticalJustification { V_J_START, V_J_END, V_J_CENTRE, V_J_JUSTIFIED };

// Corresponding MHEG datatype: LineOrientation
//=============================================================
enum LineOrientation { VERTICAL, HORIZONTAL };

// Corresponding MHEG datatype: StartCorner
//=============================================================
enum StartCorner { UPPER_LEFT, UPPER_RIGHT, LOWER_LEFT, LOWER_RIGHT };

// Corresponding MHEG datatype: StreamContent
//=============================================================
enum StreamContentTag { S_C_AUDIO_CLASS_TAG, S_C_VIDEO_CLASS_TAG, S_C_RTGRAPHICS_CLASS_TAG };
union StreamContent
switch (StreamContentTag) {
    case S_C_AUDIO_CLASS_TAG:
        AudioClass  audio_class;
    case S_C_VIDEO_CLASS_TAG:
        VideoClass  video_class;
    case S_C_RTGRAPHICS_CLASS_TAG:
        RTGraphicsClass rtgraphics_class;
};

// Corresponding MHEG datatype: Storage
//=============================================================
enum Storage { MEMORY, STREAM };

// Corresponding MHEG datatype: VideoTermination
//=============================================================
enum VideoTermination { FREEZE, DISAPPEAR };

// Corresponding MHEG datatype: Orientation
//=============================================================
enum Orientation { LEFT, RIGHT, UP, DOWN };

// Corresponding MHEG datatype: SliderStyle
//=============================================================
enum SliderStyle { NORMAL, THERMOMETER, PROPORTIONAL };

// Corresponding MHEG datatype: InputType
//=============================================================
enum InputType { ALPHA, NUMERIC, ANY, LISTED };

// Corresponding MHEG datatype: ScrollType
//=============================================================
enum ScrollType { S_T_UP, S_T_DOWN, S_T_ABSOLUTE };

// Corresponding MHEG datatype: ObjectReference
//=============================================================
struct ObjectReference {
    sequence<octet> group_identifier;
    long    object_number;
};

// Corresponding MHEG datatype: ContentReference
//=============================================================
struct ContentReference {
    sequence<octet> public_identifier;
    sequence<octet> system_identifier;
```

```
};

// Corresponding MHEG datatype: GenericObjectReference
//=================================================================
enum GenericObjectReferenceTag { DIRECT_REFERENCE_TAG, INDIRECT_REFERENCE_TAG };
union GenericObjectReference
switch (GenericObjectReferenceTag) {
    case DIRECT_REFERENCE_TAG:
        ObjectReference direct_reference;
    case INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericContentReference
//=================================================================
enum GenericContentReferenceTag { G_C_R_CONTENT_REFERENCE_TAG, G_C_R_INDIRECT_REFERENCE_TAG };
union GenericContentReference
switch (GenericContentReferenceTag) {
    case G_C_R_CONTENT_REFERENCE_TAG:
        ContentReference    content_reference;
    case G_C_R_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericInteger
//=================================================================
enum GenericIntegerTag { G_I_INTEGER_VALUE_TAG, G_I_INDIRECT_REFERENCE_TAG };
union GenericInteger
switch (GenericIntegerTag) {
    case G_I_INTEGER_VALUE_TAG:
        long    integer_value;
    case G_I_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericBoolean
//=================================================================
enum GenericBooleanTag { G_B_BOOLEAN_VALUE_TAG, G_B_INDIRECT_REFERENCE_TAG };
union GenericBoolean
switch (GenericBooleanTag) {
    case G_B_BOOLEAN_VALUE_TAG:
        boolean boolean_value;
    case G_B_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: GenericOctetString
//=================================================================
enum GenericOctetStringTag { G_O_S_OCTETSTRING_VALUE_TAG, G_O_S_INDIRECT_REFERENCE_TAG };
union GenericOctetString
switch (GenericOctetStringTag) {
    case G_O_S_OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
    case G_O_S_INDIRECT_REFERENCE_TAG:
        ObjectReference indirect_reference;
};

// Corresponding MHEG datatype: Colour
//=================================================================
enum ColourTag { COLOUR_INDEX_TAG, ABSOLUTE_COLOUR_TAG };
union Colour
switch (ColourTag) {
    case COLOUR_INDEX_TAG:
        long    colour_index;
    case ABSOLUTE_COLOUR_TAG:
        sequence<octet> absolute_colour;
};

// Corresponding MHEG datatype: ContentData
//=================================================================
enum ContentDataTag { INCLUDED_CONTENT_TAG, REFERENCED_CONTENT_TAG };
union ContentData
switch (ContentDataTag) {
    case INCLUDED_CONTENT_TAG:
        sequence<octet> included_content;
    case REFERENCED_CONTENT_TAG:
        ContentReference    referenced_content;
};

// Corresponding MHEG datatype: LinkCondition
//=================================================================
struct LinkCondition {
    ObjectReference event_source;
```

```
    EventType   event_type;
    sequence<EventData,1>   event_data;
};

// Corresponding MHEG datatype: OriginalValue
//=============================================================
enum OriginalValueTag { O_V_BOOLEAN_VALUE_TAG, O_V_INTEGER_VALUE_TAG, O_V_OCTETSTRING_VALUE_TAG,
O_V_OBJECT_REFERENCE_TAG, O_V_CONTENT_REFERENCE_TAG };
union OriginalValue
switch (OriginalValueTag) {
    case O_V_BOOLEAN_VALUE_TAG:
        boolean boolean_value;
    case O_V_INTEGER_VALUE_TAG:
        long    integer_value;
    case O_V_OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
    case O_V_OBJECT_REFERENCE_TAG:
        ObjectReference object_reference;
    case O_V_CONTENT_REFERENCE_TAG:
        ContentReference    content_reference;
};

// Corresponding MHEG datatype: TokenGroupItem
//=============================================================
struct TokenGroupItem {
    Avisible    a_visible;
    sequence<ActionSlot>    action_slots;
};

// Corresponding MHEG datatype: ItemTemplate
//=============================================================
struct ItemTemplate {
    sequence<TemplateVisible>   template_visibles;
    sequence<VariableClass> template_variables;
};

// Corresponding MHEG datatype: VisibleContent
//=============================================================
enum VisibleContentTag { V_C_CONTENT_REFERENCE_TAG, V_C_NULL_TAG };
union VisibleContent
switch (VisibleContentTag) {
    case V_C_CONTENT_REFERENCE_TAG:
        ContentReference    content_reference;
};

// Corresponding MHEG datatype: VariableContent
//=============================================================
enum VariableContentTag { VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG,
VARIABLE_CONTENT_INTEGER_VALUE_TAG, VARIABLE_CONTENT_BOOLEAN_VALUE_TAG,
VARIABLE_CONTENT_OBJECT_REFERENCE_TAG, VARIABLE_CONTENT_CONTENT_REFERENCE_TAG,
VARIABLE_CONTENT_NULL_TAG };
union VariableContent
switch (VariableContentTag) {
    case VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG:
        sequence<octet> octetstring_value;
    case VARIABLE_CONTENT_INTEGER_VALUE_TAG:
        long    integer_value;
    case VARIABLE_CONTENT_BOOLEAN_VALUE_TAG:
        boolean boolean_value;
    case VARIABLE_CONTENT_OBJECT_REFERENCE_TAG:
        ObjectReference object_reference;
    case VARIABLE_CONTENT_CONTENT_REFERENCE_TAG:
        ContentReference    content_reference;
};

// Corresponding MHEG datatype: Font
//=============================================================
enum FontTag { FONT_NAME_TAG, FONT_REFERENCE_TAG };
union Font
switch (FontTag) {
    case FONT_NAME_TAG:
        sequence<octet> font_name;
    case FONT_REFERENCE_TAG:
        ObjectReference font_reference;
};

// Corresponding MHEG datatype: StreamComponent
//=============================================================
struct StreamComponent {
    long    stream_tag;
    StreamContent   stream_content;
};
```

```
// Corresponding MHEG datatype: CallActionSlot
//=================================================================
struct CallActionSlot {
    GenericObjectReference  target;
    GenericInteger  index;
};

// Corresponding MHEG datatype: CallTemplateAction
//=================================================================
struct CallTemplateAction {
    GenericObjectReference  target;
    GenericInteger  index;
};

// Corresponding MHEG datatype: CloseConnection
//=================================================================
struct CloseConnection {
    GenericObjectReference  target;
    GenericInteger  connection_tag;
};

// Corresponding MHEG datatype: GetAvailabilityStatus
//=================================================================
struct GetAvailabilityStatus {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetCursorPosition
//=================================================================
struct GetCursorPosition {
    GenericObjectReference  target;
    ObjectReference x_out;
    ObjectReference y_out;
};

// Corresponding MHEG datatype: GetEngineSupport
//=================================================================
struct GetEngineSupport {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetEntryPoint
//=================================================================
struct GetEntryPoint {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetItemContent
//=================================================================
struct GetItemContent {
    GenericObjectReference  target;
    ObjectReference content_variable;
};

// Corresponding MHEG datatype: GetRunningStatus
//=================================================================
struct GetRunningStatus {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetSelectionStatus
//=================================================================
struct GetSelectionStatus {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetSliderValue
//=================================================================
struct GetSliderValue {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: GetState
//=================================================================
struct GetState {
    GenericObjectReference  target;
    ObjectReference state_variable;
```

```
};

// Corresponding MHEG datatype: GetTextData
//=============================================================
struct GetTextData {
    GenericObjectReference  target;
    ObjectReference result;
};

// Corresponding MHEG datatype: Move
//=============================================================
struct Move {
    GenericObjectReference  target;
    GenericInteger  movement_identifier;
};

// Corresponding MHEG datatype: MoveTo
//=============================================================
struct MoveTo {
    GenericObjectReference  target;
    GenericInteger  index;
};

// Corresponding MHEG datatype: OpenConnection
//=============================================================
struct OpenConnection {
    GenericObjectReference  target;
    ObjectReference result;
    GenericOctetString  protocol;
    GenericOctetString  address;
    GenericInteger  connection_tag;
};

// Corresponding MHEG datatype: NewVisibleContent
//=============================================================
enum NewVisibleContentTag { N_V_C_NEW_CONTENT_REFERENCE_TAG, N_V_C_NULL_TAG };
union NewVisibleContent
switch (NewVisibleContentTag) {
    case N_V_C_NEW_CONTENT_REFERENCE_TAG:
        GenericContentReference new_content_reference;
};

// Corresponding MHEG datatype: NewVariableContent
//=============================================================
enum NewVariableContentTag { NEW_VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG,
NEW_VARIABLE_CONTENT_INTEGER_VALUE_TAG, NEW_VARIABLE_CONTENT_BOOLEAN_VALUE_TAG,
NEW_VARIABLE_CONTENT_OBJECT_REFERENCE_TAG, NEW_VARIABLE_CONTENT_CONTENT_REFERENCE_TAG,
NEW_VARIABLE_CONTENT_NULL_TAG };
union NewVariableContent
switch (NewVariableContentTag) {
    case NEW_VARIABLE_CONTENT_OCTETSTRING_VALUE_TAG:
        GenericOctetString  octetstring_value;
    case NEW_VARIABLE_CONTENT_INTEGER_VALUE_TAG:
        GenericInteger  integer_value;
    case NEW_VARIABLE_CONTENT_BOOLEAN_VALUE_TAG:
        GenericBoolean  boolean_value;
    case NEW_VARIABLE_CONTENT_OBJECT_REFERENCE_TAG:
        GenericObjectReference  object_reference;
    case NEW_VARIABLE_CONTENT_CONTENT_REFERENCE_TAG:
        GenericContentReference content_reference;
};

// Corresponding MHEG datatype: PutBefore
//=============================================================
struct PutBefore {
    GenericObjectReference  target;
    GenericObjectReference  reference_visible;
};

// Corresponding MHEG datatype: PutBehind
//=============================================================
struct PutBehind {
    GenericObjectReference  target;
    GenericObjectReference  reference_visible;
};

// Corresponding MHEG datatype: ReadPersistent
//=============================================================
struct ReadPersistent {
    GenericObjectReference  target;
    ObjectReference result;
    sequence<ObjectReference>   out_variables;
    GenericOctetString  in_file_name;
```

```
};

// Corresponding MHEG datatype: InParameters
//=================================================================
struct InParameters {
    sequence<GenericBoolean>    in_booleans;
    sequence<GenericInteger>    in_integer;
    sequence<GenericOctetString>    in_octetstring;
    sequence<GenericObjectReference>    in_object_reference;
    sequence<GenericContentReference>   in_content_reference;
};

// Corresponding MHEG datatype: OutParameters
//=================================================================
struct OutParameters {
    sequence<ObjectReference>   out_booleans;
    sequence<ObjectReference>   out_integer;
    sequence<ObjectReference>   out_octetstring;
    sequence<ObjectReference>   out_object_reference;
    sequence<ObjectReference>   out_content_reference;
};

// Corresponding MHEG datatype: RunSynchronous
//=================================================================
struct RunSynchronous {
    GenericObjectReference  target;
    ObjectReference result;
    InParameters    in_parameters;
    OutParameters   out_parameters;
};

// Corresponding MHEG datatype: ScaleBitmap
//=================================================================
struct ScaleBitmap {
    GenericObjectReference  target;
    GenericInteger  x_scale;
    GenericInteger  y_scale;
};

// Corresponding MHEG datatype: ScaleVideo
//=================================================================
struct ScaleVideo {
    GenericObjectReference  target;
    GenericInteger  x_scale;
    GenericInteger  y_scale;
};

// Corresponding MHEG datatype: Scroll
//=================================================================
struct Scroll {
    GenericObjectReference  target;
    ScrollType  scroll_type;
    GenericInteger  scroll_number;
};

// Corresponding MHEG datatype: EmulatedEventData
//=================================================================
enum EmulatedEventDataTag { GENERIC_BOOLEAN_TAG, GENERIC_INTEGER_TAG, GENERIC_OCTET_STRING_TAG };
union EmulatedEventData
switch (EmulatedEventDataTag) {
    case GENERIC_BOOLEAN_TAG:
        GenericBoolean  generic_boolean;
    case GENERIC_INTEGER_TAG:
        GenericInteger  generic_integer;
    case GENERIC_OCTET_STRING_TAG:
        GenericOctetString  generic_octet_string;
};

// Corresponding MHEG datatype: SetBitmapPaletteRef
//=================================================================
struct SetBitmapPaletteRef {
    GenericObjectReference  target;
    GenericObjectReference  new_bitmap_palette;
};

// Corresponding MHEG datatype: SetBoxSize
//=================================================================
struct SetBoxSize {
    GenericObjectReference  target;
    GenericInteger  x_box_size;
    GenericInteger  y_box_size;
};
```

```
// Corresponding MHEG datatype: SetCachePriority
//=============================================================
struct SetCachePriority {
    GenericObjectReference  target;
    GenericInteger  new_cache_priority;
};

// Corresponding MHEG datatype: SetCounterPosition
//=============================================================
struct SetCounterPosition {
    GenericObjectReference  target;
    GenericInteger  new_counter_position;
};

// Corresponding MHEG datatype: SetCounterTrigger
//=============================================================
struct SetCounterTrigger {
    GenericObjectReference  target;
    GenericInteger  trigger_identifier;
    sequence<GenericInteger,1>  new_counter_value;
};

// Corresponding MHEG datatype: SetCursorPosition
//=============================================================
struct SetCursorPosition {
    GenericObjectReference  target;
    GenericInteger  x_cursor;
    GenericInteger  y_cursor;
};

// Corresponding MHEG datatype: SetCursorShape
//=============================================================
struct SetCursorShape {
    GenericObjectReference  target;
    sequence<GenericObjectReference,1>  new_cursor_shape;
};

// Corresponding MHEG datatype: NewContent
//=============================================================
enum NewContentTag { NEW_INCLUDED_CONTENT_TAG, NEW_REFERENCED_CONTENT_TAG };
union NewContent
switch (NewContentTag) {
    case NEW_INCLUDED_CONTENT_TAG:
        GenericOctetString  new_included_content;
    case NEW_REFERENCED_CONTENT_TAG:
        GenericContentReference new_referenced_content;
};

// Corresponding MHEG datatype: SetEntryPoint
//=============================================================
struct SetEntryPoint {
    GenericObjectReference  target;
    GenericInteger  new_entry_point;
};

// Corresponding MHEG datatype: NewColour
//=============================================================
enum NewColourTag { N_C_COLOUR_INDEX_TAG, N_C_ABSOLUTE_COLOUR_TAG };
union NewColour
switch (NewColourTag) {
    case N_C_COLOUR_INDEX_TAG:
        GenericInteger  colour_index;
    case N_C_ABSOLUTE_COLOUR_TAG:
        GenericOctetString  absolute_colour;
};

// Corresponding MHEG datatype: NewFontRef
//=============================================================
enum NewFontRefTag { N_F_R_FONT_NAME_TAG, N_F_R_FONT_REFERENCE_TAG };
union NewFontRef
switch (NewFontRefTag) {
    case N_F_R_FONT_NAME_TAG:
        GenericOctetString  font_name;
    case N_F_R_FONT_REFERENCE_TAG:
        ObjectReference font_reference;
};

// Corresponding MHEG datatype: SetHighlightStatus
//=============================================================
struct SetHighlightStatus {
    GenericObjectReference  target;
    GenericBoolean  new_highlight_status;
};
```

```
// Corresponding MHEG datatype: SetInteractiblePalette
//=============================================================
struct SetInteractiblePalette {
    GenericObjectReference  target;
    GenericObjectReference  new_interactible_palette;
};

// Corresponding MHEG datatype: SetInteractionStatus
//=============================================================
struct SetInteractionStatus {
    GenericObjectReference  target;
    GenericBoolean  new_interaction_status;
};

// Corresponding MHEG datatype: SetItemContent
//=============================================================
struct SetItemContent {
    GenericObjectReference  target;
    GenericOctetString  new_items_content;
};

// Corresponding MHEG datatype: SetLabel
//=============================================================
struct SetLabel {
    GenericObjectReference  target;
    GenericOctetString  new_label;
};

// Corresponding MHEG datatype: SetLineartPalette
//=============================================================
struct SetLineartPalette {
    GenericObjectReference  target;
    GenericObjectReference  new_lineart_palette;
};

// Corresponding MHEG datatype: SetLineColour
//=============================================================
struct SetLineColour {
    GenericObjectReference  target;
    NewColour   new_line_colour;
};

// Corresponding MHEG datatype: SetLineWidth
//=============================================================
struct SetLineWidth {
    GenericObjectReference  target;
    long    line_width;
};

// Corresponding MHEG datatype: SetOverwriteMode
//=============================================================
struct SetOverwriteMode {
    GenericObjectReference  target;
    GenericBoolean  new_overwrite_mode;
};

// Corresponding MHEG datatype: SetPortion
//=============================================================
struct SetPortion {
    GenericObjectReference  target;
    GenericInteger  new_portion;
};

// Corresponding MHEG datatype: SetPosition
//=============================================================
struct SetPosition {
    GenericObjectReference  target;
    GenericInteger  new_x_position;
    GenericInteger  new_y_position;
};

// Corresponding MHEG datatype: SetSliderValue
//=============================================================
struct SetSliderValue {
    GenericObjectReference  target;
    GenericInteger  new_slider_value;
};

// Corresponding MHEG datatype: Rational
//=============================================================
struct Rational {
    GenericInteger  nominator;
```

```
    sequence<GenericInteger,1>  denominator;
};

// Corresponding MHEG datatype: SetState
//=============================================================
struct SetState {
    GenericObjectReference  target;
    GenericOctetString  new_state;
};

// Corresponding MHEG datatype: SetTextPaletteRef
//=============================================================
struct SetTextPaletteRef {
    GenericObjectReference  target;
    GenericObjectReference  new_text_palette;
};

// Corresponding MHEG datatype: SetTimer
//=============================================================
struct SetTimer {
    GenericObjectReference  target;
    GenericInteger  timer_id;
    sequence<GenericInteger,1>  timer_value;
};

// Corresponding MHEG datatype: SetTransparency
//=============================================================
struct SetTransparency {
    GenericObjectReference  target;
    GenericInteger  new_transparency;
};

// Corresponding MHEG datatype: NewVariableValue
//=============================================================
enum NewVariableValueTag { N_V_V_GENERIC_INTEGER_TAG, N_V_V_GENERIC_BOOLEAN_TAG,
N_V_V_GENERIC_OCTET_STRING_TAG, N_V_V_GENERIC_OBJECT_REFERENCE_TAG,
N_V_V_GENERIC_CONTENT_REFERENCE_TAG };
union NewVariableValue
switch (NewVariableValueTag) {
    case N_V_V_GENERIC_INTEGER_TAG:
        GenericInteger  generic_integer;
    case N_V_V_GENERIC_BOOLEAN_TAG:
        GenericBoolean  generic_boolean;
    case N_V_V_GENERIC_OCTET_STRING_TAG:
        GenericOctetString  generic_octet_string;
    case N_V_V_GENERIC_OBJECT_REFERENCE_TAG:
        GenericObjectReference  generic_object_reference;
    case N_V_V_GENERIC_CONTENT_REFERENCE_TAG:
        GenericContentReference generic_content_reference;
};

// Corresponding MHEG datatype: SetVolume
//=============================================================
struct SetVolume {
    GenericObjectReference  target;
    GenericInteger  new_volume;
};

// Corresponding MHEG datatype: Step
//=============================================================
struct Step {
    GenericObjectReference  target;
    GenericInteger  nb_of_steps;
};

// Corresponding MHEG datatype: StorePersistent
//=============================================================
struct StorePersistent {
    GenericObjectReference  target;
    ObjectReference result;
    sequence<ObjectReference>   in_variables;
    GenericOctetString  out_file_name;
};

// Corresponding MHEG datatype: ComparisonValue
//=============================================================
enum ComparisonValueTag { C_V_GENERIC_INTEGER_TAG, C_V_GENERIC_BOOLEAN_TAG,
C_V_GENERIC_OCTET_STRING_TAG, C_V_GENERIC_OBJECT_REFERENCE_TAG, C_V_GENERIC_CONTENT_REFERENCE_TAG
};
union ComparisonValue
switch (ComparisonValueTag) {
    case C_V_GENERIC_INTEGER_TAG:
        GenericInteger  generic_integer;
```

```
    case C_V_GENERIC_BOOLEAN_TAG:
        GenericBoolean  generic_boolean;
    case C_V_GENERIC_OCTET_STRING_TAG:
        GenericOctetString  generic_octet_string;
    case C_V_GENERIC_OBJECT_REFERENCE_TAG:
        GenericObjectReference  generic_object_reference;
    case C_V_GENERIC_CONTENT_REFERENCE_TAG:
        GenericContentReference generic_content_reference;
};

// Corresponding MHEG datatype: TransitionTo
//===============================================================
struct TransitionTo {
    GenericObjectReference  target;
    sequence<GenericInteger,1>  connection_tag;
    GenericInteger  transition_effect;
};

// Corresponding MHEG datatype: Unplug
//===============================================================
struct Unplug {
    GenericObjectReference  target;
    sequence<GenericInteger,1>  index;
};

// Corresponding MHEG datatype: ItemData
//===============================================================
struct ItemData {
    sequence<VisibleContent>    visibles_content;
    sequence<VariableContent>   variables_content;
};

// Corresponding MHEG datatype: NewItemData
//===============================================================
struct NewItemData {
    sequence<NewVisibleContent> new_visibles_content;
    sequence<NewVariableContent>    new_variables_content;
};

// Corresponding MHEG datatype: RunAsynchronous
//===============================================================
struct RunAsynchronous {
    GenericObjectReference  target;
    ObjectReference result;
    InParameters    in_parameters;
    OutParameters   out_parameters;
};

// Corresponding MHEG datatype: SendEvent
//===============================================================
struct SendEvent {
    GenericObjectReference  target;
    ObjectReference event_source;
    EventType   event_type;
    sequence<EmulatedEventData,1>   emulated_event_data;
};

// Corresponding MHEG datatype: SetData
//===============================================================
struct SetData {
    GenericObjectReference  target;
    NewContent  new_content;
};

// Corresponding MHEG datatype: SetFillColour
//===============================================================
struct SetFillColour {
    GenericObjectReference  target;
    NewColour   new_fill_colour;
};

// Corresponding MHEG datatype: SetFontRef
//===============================================================
struct SetFontRef {
    GenericObjectReference  target;
    NewFontRef  new_font_ref;
};

// Corresponding MHEG datatype: SetSpeed
//===============================================================
struct SetSpeed {
    GenericObjectReference  target;
    Rational    new_speed;
```

```
};

// Corresponding MHEG datatype: SetVariable
//=============================================================
struct SetVariable {
    GenericObjectReference  target;
    NewVariableValue    new_variable_value;
};

// Corresponding MHEG datatype: TestVariable
//=============================================================
struct TestVariable {
    GenericObjectReference  target;
    ComparisonValue comparison_value;
    ActionClass action_true;
    ActionClass action_false;
};

// Corresponding MHEG datatype: Plug
//=============================================================
struct Plug {
    GenericObjectReference  target;
    NewItemData item_data;
    sequence<GenericInteger,1>  index;
};

// Corresponding MHEG datatype: ElementaryAction
//=============================================================
enum ElementaryActionTag { ACTIVATE_TAG, BRING_TO_FRONT_TAG, CALL_ACTION_SLOT_TAG,
CALL_TEMPLATE_ACTION_TAG, CLOSE_CONNECTION_TAG, DEACTIVATE_TAG, DESELECT_TAG, DESELECT_ITEM_TAG,
GET_AVAILABILITY_STATUS_TAG, GET_CURSOR_POSITION_TAG, GET_ENGINE_SUPPORT_TAG,
GET_ENTRY_POINT_TAG, GET_ITEM_CONTENT_TAG, GET_RUNNING_STATUS_TAG, GET_SELECTION_STATUS_TAG,
GET_SLIDER_VALUE_TAG, GET_STATE_TAG, GET_TEXT_DATA_TAG, LAUNCH_TAG, LOCK_SCREEN_TAG, MOVE_TAG,
MOVE_TO_TAG, OPEN_CONNECTION_TAG, PLUG_TAG, PRELOAD_TAG, PUT_BEFORE_TAG, PUT_BEHIND_TAG,
QUIT_TAG, READ_PERSISTENT_TAG, RUN_TAG, RUN_ASYNCHRONOUS_TAG, RUN_SYNCHRONOUS_TAG,
SCALE_BITMAP_TAG, SCALE_VIDEO_TAG, SCROLL_TAG, SELECT_TAG, SELECT_ITEM_TAG, SEND_EVENT_TAG,
SEND_TO_BACK_TAG, SET_BITMAP_PALETTE_REF_TAG, SET_BOX_SIZE_TAG, SET_CACHE_PRIORITY_TAG,
SET_COUNTER_POSITION_TAG, SET_COUNTER_TRIGGER_TAG, SET_CURSOR_POSITION_TAG, SET_CURSOR_SHAPE_TAG,
SET_DATA_TAG, SET_ENTRY_POINT_TAG, SET_FILL_COLOUR_TAG, SET_FONT_REF_TAG,
SET_HIGHLIGHT_STATUS_TAG, SET_INTERACTIBLE_PALETTE_TAG, SET_INTERACTION_STATUS_TAG,
SET_ITEM_CONTENT_TAG, SET_LABEL_TAG, SET_LINEART_PALETTE_TAG, SET_LINE_COLOUR_TAG,
SET_LINE_WIDTH_TAG, SET_OVERWRITE_MODE_TAG, SET_PORTION_TAG, SET_POSITION_TAG,
SET_SLIDER_VALUE_TAG, SET_SPEED_TAG, SET_STATE_TAG, SET_TEXT_PALETTE_REF_TAG, SET_TIMER_TAG,
SET_TRANSPARENCY_TAG, SET_VARIABLE_TAG, SET_VOLUME_TAG, SPAWN_TAG, STOP_TAG, STEP_TAG,
STORE_PERSISTENT_TAG, TEST_VARIABLE_TAG, TOGGLE_TAG, TOGGLE_ITEM_TAG, TRANSITION_TO_TAG,
UNLOAD_TAG, UNLOCK_SCREEN_TAG, UNPLUG_TAG };
union ElementaryAction
switch (ElementaryActionTag) {
    case ACTIVATE_TAG:
        GenericObjectReference  activate;
    case BRING_TO_FRONT_TAG:
        GenericObjectReference  bring_to_front;
    case CALL_ACTION_SLOT_TAG:
        CallActionSlot  call_action_slot;
    case CALL_TEMPLATE_ACTION_TAG:
        CallTemplateAction  call_template_action;
    case CLOSE_CONNECTION_TAG:
        CloseConnection close_connection;
    case DEACTIVATE_TAG:
        GenericObjectReference  deactivate;
    case DESELECT_TAG:
        GenericObjectReference  deselect;
    case DESELECT_ITEM_TAG:
        GenericObjectReference  deselect_item;
    case GET_AVAILABILITY_STATUS_TAG:
        GetAvailabilityStatus   get_availability_status;
    case GET_CURSOR_POSITION_TAG:
        GetCursorPosition   get_cursor_position;
    case GET_ENGINE_SUPPORT_TAG:
        GetEngineSupport    get_engine_support;
    case GET_ENTRY_POINT_TAG:
        GetEntryPoint   get_entry_point;
    case GET_ITEM_CONTENT_TAG:
        GetItemContent  get_item_content;
    case GET_RUNNING_STATUS_TAG:
        GetRunningStatus    get_running_status;
    case GET_SELECTION_STATUS_TAG:
        GetSelectionStatus  get_selection_status;
    case GET_SLIDER_VALUE_TAG:
        GetSliderValue  get_slider_value;
    case GET_STATE_TAG:
        GetState    get_state;
    case GET_TEXT_DATA_TAG:
```

```
    GetTextData get_text_data;
case LAUNCH_TAG:
    GenericObjectReference  launch;
case LOCK_SCREEN_TAG:
    GenericObjectReference  lock_screen;
case MOVE_TAG:
    Move    the_move;
case MOVE_TO_TAG:
    MoveTo  move_to;
case OPEN_CONNECTION_TAG:
    OpenConnection  open_connection;
case PLUG_TAG:
    Plug    the_plug;
case PRELOAD_TAG:
    GenericObjectReference  preload;
case PUT_BEFORE_TAG:
    PutBefore   put_before;
case PUT_BEHIND_TAG:
    PutBehind   put_behind;
case QUIT_TAG:
    GenericObjectReference  quit;
case READ_PERSISTENT_TAG:
    ReadPersistent  read_persistent;
case RUN_TAG:
    GenericObjectReference  run;
case RUN_ASYNCHRONOUS_TAG:
    RunAsynchronous run_asynchronous;
case RUN_SYNCHRONOUS_TAG:
    RunSynchronous  run_synchronous;
case SCALE_BITMAP_TAG:
    ScaleBitmap scale_bitmap;
case SCALE_VIDEO_TAG:
    ScaleVideo  scale_video;
case SCROLL_TAG:
    Scroll  the_scroll;
case SELECT_TAG:
    GenericObjectReference  select;
case SELECT_ITEM_TAG:
    GenericObjectReference  select_item;
case SEND_EVENT_TAG:
    SendEvent   send_event;
case SEND_TO_BACK_TAG:
    GenericObjectReference  send_to_back;
case SET_BITMAP_PALETTE_REF_TAG:
    SetBitmapPaletteRef set_bitmap_palette_ref;
case SET_BOX_SIZE_TAG:
    SetBoxSize  set_box_size;
case SET_CACHE_PRIORITY_TAG:
    SetCachePriority    set_cache_priority;
case SET_COUNTER_POSITION_TAG:
    SetCounterPosition  set_counter_position;
case SET_COUNTER_TRIGGER_TAG:
    SetCounterTrigger   set_counter_trigger;
case SET_CURSOR_POSITION_TAG:
    SetCursorPosition   set_cursor_position;
case SET_CURSOR_SHAPE_TAG:
    SetCursorShape  set_cursor_shape;
case SET_DATA_TAG:
    SetData set_data;
case SET_ENTRY_POINT_TAG:
    SetEntryPoint   set_entry_point;
case SET_FILL_COLOUR_TAG:
    SetFillColour   set_fill_colour;
case SET_FONT_REF_TAG:
    SetFontRef  set_font_ref;
case SET_HIGHLIGHT_STATUS_TAG:
    SetHighlightStatus  set_highlight_status;
case SET_INTERACTIBLE_PALETTE_TAG:
    SetInteractiblePalette  set_interactible_palette;
case SET_INTERACTION_STATUS_TAG:
    SetInteractionStatus    set_interaction_status;
case SET_ITEM_CONTENT_TAG:
    SetItemContent  set_item_content;
case SET_LABEL_TAG:
    SetLabel    set_label;
case SET_LINEART_PALETTE_TAG:
    SetLineartPalette   set_lineart_palette;
case SET_LINE_COLOUR_TAG:
    SetLineColour   set_line_colour;
case SET_LINE_WIDTH_TAG:
    SetLineWidth    set_line_width;
case SET_OVERWRITE_MODE_TAG:
    SetOverwriteMode    set_overwrite_mode;
```

```
    case SET_PORTION_TAG:
        SetPortion  set_portion;
    case SET_POSITION_TAG:
        SetPosition set_position;
    case SET_SLIDER_VALUE_TAG:
        SetSliderValue  set_slider_value;
    case SET_SPEED_TAG:
        SetSpeed    set_speed;
    case SET_STATE_TAG:
        SetState    set_state;
    case SET_TEXT_PALETTE_REF_TAG:
        SetTextPaletteRef   set_text_palette_ref;
    case SET_TIMER_TAG:
        SetTimer    set_timer;
    case SET_TRANSPARENCY_TAG:
        SetTransparency set_transparency;
    case SET_VARIABLE_TAG:
        SetVariable set_variable;
    case SET_VOLUME_TAG:
        SetVolume   set_volume;
    case SPAWN_TAG:
        GenericObjectReference  spawn;
    case STOP_TAG:
        GenericObjectReference  stop;
    case STEP_TAG:
        Step    the_step;
    case STORE_PERSISTENT_TAG:
        StorePersistent store_persistent;
    case TEST_VARIABLE_TAG:
        TestVariable    test_variable;
    case TOGGLE_TAG:
        GenericObjectReference  toggle;
    case TOGGLE_ITEM_TAG:
        GenericObjectReference  toggle_item;
    case TRANSITION_TO_TAG:
        TransitionTo    transition_to;
    case UNLOAD_TAG:
        GenericObjectReference  unload;
    case UNLOCK_SCREEN_TAG:
        GenericObjectReference  unlock_screen;
    case UNPLUG_TAG:
        Unplug  the_unplug;
};

// Corresponding MHEG datatype: RootClass
//===============================================================
interface RootClass {
    attribute   ObjectReference object_identifier;
    attribute   sequence<boolean,1> no_caching;
};

// Corresponding MHEG datatype: GroupClass
//===============================================================
interface GroupClass : RootClass {
    attribute   sequence<long,1>    standard_version;
    attribute   sequence<octet> object_information;
    attribute   sequence<ActionClass,1> on_start_up;
    attribute   sequence<ActionClass,1> on_close_down;
    attribute   sequence<long,1>    cache_priority;
    attribute   sequence<Item>  items;
};

// Corresponding MHEG datatype: ApplicationClass
//===============================================================
interface ApplicationClass : GroupClass {
    attribute   ObjectIdentifier    object_identifier;
    attribute   sequence<ActionClass,1> on_spawn_close_down;
    attribute   sequence<ActionClass,1> on_restart;
};

// Corresponding MHEG datatype: SceneClass
//===============================================================
interface SceneClass : GroupClass {
    attribute   ObjectIdentifier    object_identifier;
    attribute   long    input_event_register;
    attribute   SceneCoordinateSystem   scene_coordinate_system;
    attribute   sequence<AspectRatio,1> aspect_ratio;
    attribute   sequence<boolean,1> moving_cursor;
    attribute   sequence<NextScene> next_scenes;
};

// Corresponding MHEG datatype: IngredientClass
//===============================================================
```

```
interface IngredientClass : RootClass {
    attribute   sequence<boolean,1> initially_active;
    attribute   sequence<ContentHook,1> content_hook;
    attribute   sequence<ContentData,1> content_data;
    attribute   sequence<long,1>    content_size;
    attribute   sequence<boolean,1> shared;
};

// Corresponding MHEG datatype: LinkClass
//=================================================================
interface LinkClass : IngredientClass {
    attribute   LinkCondition   link_condition;
    attribute   ActionClass link_effect;
};

// Corresponding MHEG datatype: ProcedureClass
//=================================================================
interface ProcedureClass : IngredientClass {
    attribute   ProcedureType   procedure_type;
    attribute   sequence<octet> procedure_name;
    attribute   sequence<long,1>    procedure_connection_tag;
};

// Corresponding MHEG datatype: PaletteClass
//=================================================================
interface PaletteClass : IngredientClass {
};

// Corresponding MHEG datatype: FontClass
//=================================================================
interface FontClass : IngredientClass {
};

// Corresponding MHEG datatype: CursorShapeClass
//=================================================================
interface CursorShapeClass : IngredientClass {
};

// Corresponding MHEG datatype: VariableClass
//=================================================================
interface VariableClass : IngredientClass {
    attribute   OriginalValue   original_value;
};

// Corresponding MHEG datatype: PresentableClass
//=================================================================
interface PresentableClass : IngredientClass {
};

// Corresponding MHEG datatype: TokenManagerClass
//=================================================================
interface TokenManagerClass {
    attribute   sequence<Movement>  movement_table;
};

// Corresponding MHEG datatype: TokenGroupClass
//=================================================================
interface TokenGroupClass : PresentableClass, TokenManagerClass {
    attribute   sequence<TokenGroupItem>    token_group_items;
};

// Corresponding MHEG datatype: TemplateGroupClass
//=================================================================
interface TemplateGroupClass : PresentableClass, TokenManagerClass {
    attribute   sequence<boolean,1> multiple_selection;
    attribute   ItemTemplate    item_template;
    attribute   sequence<ItemData>  items_data;
    attribute   sequence<ActionClass,1> on_get_token;
    attribute   sequence<ActionClass,1> on_lose_token;
    attribute   sequence<ActionClass,1> on_toggle;
    attribute   sequence<ActionClass,1> on_select;
    attribute   sequence<ActionClass,1> on_deselect;
    attribute   sequence<TemplateAction>    template_actions;
    attribute   sequence<CellPosition>  cell_positions;
};

// Corresponding MHEG datatype: ListClass
//=================================================================
interface ListClass : TemplateGroupClass {
    attribute   sequence<ActionClass,1> on_visible;
    attribute   sequence<ActionClass,1> on_invisible;
    attribute   sequence<ActionClass,1> on_at_first_position;
    attribute   sequence<ActionClass,1> on_leave_first_position;
```

```
    attribute   sequence<ActionClass,1> on_at_last_position;
    attribute   sequence<ActionClass,1> on_leave_last_position;
    attribute   sequence<boolean,1> fixed_token;
    attribute   sequence<boolean,1> wrap_around;
};

// Corresponding MHEG datatype: VisibleClass
//=============================================================
interface VisibleClass : PresentableClass {
    attribute   OriginalBoxSize original_box_size;
    attribute   sequence<OriginalPosition,1>    original_position;
};

// Corresponding MHEG datatype: BitmapClass
//=============================================================
interface BitmapClass : VisibleClass {
    attribute   sequence<ObjectReference,1> bitmap_palette_ref;
    attribute   sequence<DropOutColour,1>   drop_out_colour;
    attribute   sequence<long,1>    transparency;
};

// Corresponding MHEG datatype: LineArtClass
//=============================================================
interface LineArtClass : VisibleClass {
    attribute   sequence<long,1>    line_width;
    attribute   sequence<LineStyle,1>   line_style;
    attribute   sequence<ObjectReference,1> lineart_palette_ref;
    attribute   Colour  ref_line_colour;
    attribute   sequence<Colour,1>  ref_fill_colour;
};

// Corresponding MHEG datatype: RectangleClass
//=============================================================
interface RectangleClass : LineArtClass {
};

// Corresponding MHEG datatype: TextClass
//=============================================================
interface TextClass : VisibleClass {
    attribute   sequence<Font,1>    the_font;
    attribute   sequence<octet> font_attributes;
    attribute   sequence<Colour,1>  text_colour;
    attribute   sequence<Colour,1>  background_colour;
    attribute   sequence<ObjectReference,1> text_palette_ref;
    attribute   sequence<long,1>    character_set;
    attribute   sequence<HorizontalJustification,1> horizontal_justification;
    attribute   sequence<VerticalJustification,1>   vertical_justification;
    attribute   sequence<LineOrientation,1> line_orientation;
    attribute   sequence<StartCorner,1> start_corner;
    attribute   sequence<boolean,1> text_wrapping;
};

// Corresponding MHEG datatype: StreamClass
//=============================================================
interface StreamClass : PresentableClass {
    attribute   sequence<StreamComponent>   multiplex;
    attribute   sequence<Storage,1> the_storage;
    attribute   sequence<long,1>    looping;
};

// Corresponding MHEG datatype: AudioClass
//=============================================================
interface AudioClass : PresentableClass {
    attribute   sequence<long,1>    original_volume;
};

// Corresponding MHEG datatype: VideoClass
//=============================================================
interface VideoClass : VisibleClass {
    attribute   sequence<VideoTermination,1>    termination;
};

// Corresponding MHEG datatype: RTGraphicsClass
//=============================================================
interface RTGraphicsClass : VisibleClass {
};

// Corresponding MHEG datatype: InteractibleClass
//=============================================================
interface InteractibleClass {
    attribute   sequence<boolean,1> engine_resp;
    attribute   sequence<Colour,1>  highlight_ref_colour;
    attribute   sequence<ObjectReference,1> interactible_palette_ref;
```

```
};

// Corresponding MHEG datatype: SliderClass
//===============================================================
interface SliderClass : VisibleClass, InteractibleClass {
    attribute   Orientation the_orientation;
    attribute   sequence<long,1>    initial_value;
    attribute   sequence<long,1>    initial_portion;
    attribute   sequence<long,1>    min_value;
    attribute   long    max_value;
    attribute   sequence<long,1>    step_size;
    attribute   sequence<SliderStyle,1> slider_style;
    attribute   Colour  slider_colour1;
    attribute   Colour  slider_colour2;
};

// Corresponding MHEG datatype: EntryFieldClass
//===============================================================
interface EntryFieldClass : TextClass, InteractibleClass {
    attribute   sequence<InputType,1>   input_type;
    attribute   sequence<octet> char_list;
    attribute   sequence<boolean,1> obscured_input;
    attribute   sequence<long,1>    max_length;
};

// Corresponding MHEG datatype: HypertextClass
//===============================================================
interface HypertextClass : TextClass, InteractibleClass {
};

// Corresponding MHEG datatype: ButtonClass
//===============================================================
interface ButtonClass : VisibleClass, InteractibleClass {
    attribute   sequence<Colour,1>  ref_colour1;
    attribute   sequence<Colour,1>  ref_colour2;
};

// Corresponding MHEG datatype: HotspotClass
//===============================================================
interface HotspotClass : ButtonClass {
};

// Corresponding MHEG datatype: PushButtonClass
//===============================================================
interface PushButtonClass : ButtonClass {
    attribute   sequence<octet> label;
};

// Corresponding MHEG datatype: SwitchButtonClass
//===============================================================
interface SwitchButtonClass : ButtonClass {
    attribute   sequence<octet> label;
};

// Corresponding MHEG datatype: ActionClass
//===============================================================
interface ActionClass {
    attribute   sequence<ElementaryAction>  elementary_actions;
};

};
```

History

Document history			
July 1996	Public Enquiry	PE 110:	1996-07-22 to 1996-11-15