

Information technology — Coding of multimedia and hypermedia information

Part 5. Support for base-level interactive applications

ICS 35.040

National foreword

This British Standard reproduces verbatim ISO/IEC 13522-5 : 1997 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee IST/37, Coding of picture, audio, multimedia and hypermedia information, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled 'International Standards Correspondence Index', or using the 'Find' facility of the BSI Standards Electronic Catalogue.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to x, pages 1 to 239 and a back cover.

This British Standard, having been prepared under the direction of the DISC Board, was published under the authority of the Standards Board and comes into effect on 15 July 1997

© BSI 1997

ISBN 0 580 27584 1

Amendments issued since publication

Amd. No.	Date	Text affected

INTERNATIONAL STANDARD

ISO/IEC
13522-5

First edition
1997-04-15

Information technology — Coding of multimedia and hypermedia information —

Part 5:

Support for base-level interactive applications

*Technologies de l'information — Codage de l'information multimédia et
hypermédia —*

Partie 5: Support pour applications interactives de niveau fondamental



Reference number
ISO/IEC 13522-5:1997(E)

AVA Documents
Document Sharing Hub

Contents

1 Scope	1
1.1 Specificity of the scope	1
1.2 Issues outside the scope of this part of ISO/IEC 13522	1
2 Normative References	1
3 Terms and definitions	2
4 Conformance	4
4.1 Conformance of MHEG-5 objects	4
4.2 Conformance of MHEG-5 engines	4
5 Overview of the MHEG-5 Classes	8
5.1 Root	9
5.2 Group	9
5.3 Application	9
5.4 Scene	9
5.5 Ingredient	10
5.6 Link	10
5.7 Action	10
5.8 Program	10
5.9 Palette, Font, and CursorShape	11
5.10 Variable	11
5.11 Presentable	11
5.12 TokenGroup	12
5.13 ListGroup	12
5.14 Stream	12
5.15 Audio	12
5.16 Interactable	12
5.17 Visible	13
6 Structure of this part of ISO/IEC 13522	14
7 Notations	15
7.1 Attributes	15
7.2 Events	15
7.3 Internal behaviours	16
7.4 Effect of MHEG-5 actions	16
7.5 Formal description	16
8 Root Class	17
8.1 Attributes	17
8.2 Events	18

Descriptors: data processing, video data, audio data, graphic data processing, image processing, information interchange, communication procedure, coding (data conversion), coded representation.

8.3 Internal behaviours	18
8.4 Effect of MHEG-5 actions	20
8.5 Formal description	20
9 Group Class	21
9.1 Attributes	21
9.2 Events	22
9.3 Internal behaviours	22
9.4 Effect of MHEG-5 actions	23
9.5 Formal description	23
10 Application Class	25
10.1 Attributes	25
10.2 Events	28
10.3 Internal behaviours	28
10.4 Effect of MHEG-5 actions	28
10.5 Formal description	34
11 Scene Class	35
11.1 Attributes	35
11.2 Events	36
11.3 Internal behaviours	37
11.4 Effect of MHEG-5 actions	37
11.5 Formal description	41
12 Ingredient Class	42
12.1 Attributes	42
12.2 Events	44
12.3 Internal behaviours	44
12.4 Effect of MHEG-5 actions	44
12.5 Formal description	47
13 Link Class	48
13.1 Attributes	48
13.2 Events	49
13.3 Internal behaviours	49
13.4 Effect of MHEG-5 actions	49
13.5 Formal description	50
14 Program Class	51
14.1 Attributes	51
14.2 Events	51
14.3 Internal behaviours	52
14.4 Effect of MHEG-5 actions	52
14.5 Formal description	54
15 ResidentProgram Class	55
15.1 Attributes	55
15.2 Events	55

15.3 Internal behaviours	55
15.4 Effect of MHEG-5 actions	55
15.5 Formal description	55
16 RemoteProgram Class	56
16.1 Attributes	56
16.2 Events	56
16.3 Internal behaviours	56
16.4 Effect of MHEG-5 actions	57
16.5 Formal description	57
17 InterchangedProgram Class	58
17.1 Attributes	58
17.2 Events	58
17.3 Internal behaviours	58
17.4 Effect of MHEG-5 actions	58
17.5 Formal description	58
18 Palette Class	59
18.1 Attributes	59
18.2 Events	59
18.3 Internal behaviours	59
18.4 Effect of MHEG-5 actions	59
18.5 Formal description	59
19 Font Class	60
19.1 Attributes	60
19.2 Events	60
19.3 Internal behaviours	60
19.4 Effect of MHEG-5 actions	60
19.5 Formal description	60
20 CursorShape Class	61
20.1 Attributes	61
20.2 Events	61
20.3 Internal behaviours	61
20.4 Effect of MHEG-5 actions	61
20.5 Formal description	61
21 Variable Class	62
21.1 Attributes	62
21.2 Events	62
21.3 Internal behaviours	63
21.4 Effect of MHEG-5 actions	63
21.5 Formal description	64
22 BooleanVariable Class	65
22.1 Attributes	65
22.2 Events	65

22.3 Internal behaviours	65
22.4 Effect of MHEG-5 actions	65
22.5 Formal description	65
23 IntegerVariable Class	66
23.1 Attributes	66
23.2 Events	66
23.3 Internal behaviours	66
23.4 Effect of MHEG-5 actions	66
23.5 Formal description	68
24 OctetStringVariable Class	69
24.1 Attributes	69
24.2 Events	69
24.3 Internal behaviours	69
24.4 Effect of MHEG-5 actions	69
24.5 Formal description	70
25 ObjectRefVariable Class	71
25.1 Attributes	71
25.2 Events	71
25.3 Internal behaviours	71
25.4 Effect of MHEG-5 actions	71
25.5 Formal description	71
26 ContentRefVariable Class	72
26.1 Attributes	72
26.2 Events	72
26.3 Internal behaviours	72
26.4 Effect of MHEG-5 actions	72
26.5 Formal description	72
27 Presentable Class	73
27.1 Attributes	73
27.2 Events	73
27.3 Internal behaviours	73
27.4 Effect of MHEG-5 actions	73
27.5 Formal description	74
28 TokenManager Class	75
28.1 Attributes	75
28.2 Events	76
28.3 Internal behaviours	76
28.4 Effect of MHEG-5 actions	77
28.5 Formal description	78
29 TokenGroup Class	79
29.1 Attributes	79
29.2 Events	79

29.3 Internal behaviours	80
29.4 Effect of MHEG-5 actions	80
29.5 Formal description	80
30 ListGroup Class	82
30.1 Attributes	82
30.2 Events	85
30.3 Internal behaviours	86
30.4 Effect of MHEG-5 actions	87
30.5 Formal description	92
31 Visible Class	93
31.1 Attributes	93
31.2 Events	94
31.3 Internal behaviours	94
31.4 Effect of MHEG-5 actions	95
31.5 Formal description	99
32 Bitmap Class	100
32.1 Attributes	100
32.2 Events	101
32.3 Internal behaviours	101
32.4 Effect of MHEG-5 actions	101
32.5 Formal description	101
33 LineArt Class	103
33.1 Attributes	103
33.2 Events	105
33.3 Internal behaviours	105
33.4 Effect of MHEG-5 actions	105
33.5 Formal description	107
34 Rectangle Class	108
34.1 Attributes	108
34.2 Events	108
34.3 Internal behaviours	108
34.4 Effect of MHEG-5 actions	108
34.5 Formal description	109
35 DynamicLineArt Class	110
35.1 Attributes	110
35.2 Events	110
35.3 Internal Behaviours	110
35.4 Effect of MHEG-5 Actions	110
35.5 Formal description	116
36 Text Class	117
36.1 Attributes	117
36.2 Events	120

36.3 Internal behaviours	120
36.4 Effect of MHEG-5 actions	120
36.5 Formal description	121
37 Stream Class	123
37.1 Attributes	123
37.2 Events	125
37.3 Internal behaviours	126
37.4 Effect of MHEG-5 actions	126
37.5 Formal description	129
38 Audio Class	130
38.1 Attributes	130
38.2 Events	131
38.3 Internal behaviours	131
38.4 Effect of MHEG-5 actions	131
38.5 Formal description	132
39 Video Class	133
39.1 Attributes	133
39.2 Events	133
39.3 Internal behaviours	133
39.4 Effect of MHEG-5 actions	134
39.5 Formal description	134
40 RTGraphics Class	135
40.1 Attributes	135
40.2 Events	135
40.3 Internal behaviours	135
40.4 Effect of MHEG-5 actions	136
40.5 Formal description	136
41 Interactable Class	137
41.1 Attributes	137
41.2 Events	138
41.3 Internal behaviours	139
41.4 Effect of MHEG-5 actions	139
41.5 Formal description	141
42 Slider Class	142
42.1 Attributes	142
42.2 Events	144
42.3 Internal behaviour	144
42.4 Effect of MHEG-5 actions	145
42.5 Formal description	147
43 EntryField Class	148
43.1 Attributes	148
43.2 Events	149

43.3 Internal behaviours	150
43.4 Effect of MHEG-5 actions	150
43.5 Formal description	151
44 HyperText Class	153
44.1 Attributes	153
44.2 Events	153
44.3 Internal behaviours	153
44.4 Effect of MHEG-5 actions	154
44.5 Formal description	154
45 Button Class	155
45.1 Attributes	155
45.2 Events	156
45.3 Internal behaviours	156
45.4 Effect of MHEG-5 actions	156
45.5 Formal description	157
46 Hotspot Class	158
46.1 Attributes	158
46.2 Events	158
46.3 Internal behaviours	158
46.4 Effect of MHEG-5 actions	158
46.5 Formal description	159
47 PushButton Class	160
47.1 Attributes	160
47.2 Events	161
47.3 Internal behaviours	161
47.4 Effect of MHEG-5 actions	161
47.5 Formal description	162
48 SwitchButton Class	163
48.1 Attributes	163
48.2 Events	163
48.3 Internal behaviours	163
48.4 Effect of MHEG-5 actions	163
48.5 Formal description	165
49 Action Class	166
49.1 Attributes	166
49.2 Own internal attributes	166
49.3 Formal description	166
50 Referencing Objects, Content, Values, Colour and XYPosition	168
50.1 ObjectReference	168
50.2 ContentReference	168
50.3 GenericObjectReference	168
50.4 GenericContentReference	168

50.5 GenericInteger	169
50.6 GenericBoolean	169
50.7 GenericOctetString	169
50.8 Colour	169
50.9 XYPosition	169
50.10 Resolution of generic values	170
51 Referencing MHEG-5 Objects	171
52 Name Spaces, RemoteProgram Calls and Connections	172
53 Event Handling.....	173
53.1 Types of events	173
53.2 Synchronous events and asynchronous events	173
53.3 Event handling and Links	174
53.4 User input	174
53.5 User interaction	175
53.6 Cursor Events	175
53.7 Error Handling	175
54 Rendering Visibles	175
54.1 Coordinate system	175
54.2 Bounding box	175
54.3 Display stack	176
54.4 Transparent objects	176
54.5 Pixel aspect ratio	177
Annex A	178
Annex B	212
B.1 General Definitions	212
B.2 Definitions of Symbols	213
B.3 Terminal Symbols	213
B.4 MHEG-5 Object Definitions	215
Annex C	235
Annex D	236
D.1 Object interchange format	236
D.2 Set of classes	236
D.3 Set of features	236
D.4 Content data encoding	236
D.5 UserInput registers	237
D.6 Semantic constraints on the MHEG-5 applications	238
D.7 EngineEvent	238
D.8 GetEngineSupport	239
D.9 Protocol mapping and external interaction	239

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialised system for worldwide standardisation. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting vote.

International Standard ISO/IEC 13522-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 13522 consists of the following parts, under the general title *Information technology – Coding of multimedia and hypermedia information*:

- *Part 1: MHEG object representation – Base notation (ASN.1)*
- *Part 3: MHEG script interchange representation*
- *Part 4: MHEG registration procedure*
- *Part 5: Support for base-level interactive applications*
- *Part 6: Support for enhanced interactive applications*

Annexes A and B form an integral part of this part of ISO/IEC 13522. Annexes C and D are for information only.

Information technology — Coding of multimedia and hypermedia information —

Part 5:

Support for base-level interactive applications

1 Scope

This part of ISO/IEC 13522 specifies semantics and final-form interchange syntax for MHEG-5 objects, based on concepts defined in ISO/IEC 13522-1. These objects are intended for use in the domain of simple client/server interactive multimedia applications, e.g. (Near) Video on Demand applications, navigation and browsing applications.

1.1 Specificity of the scope

Since it is expected that this part of ISO/IEC 13522 be used for interoperability of applications across platforms, the scope focuses on a specific and precise definition of MHEG-5 classes. This part of ISO/IEC 13522 recognises the semantics implied by the specification of the MHEG-5 objects and by interpretation of MHEG-5 behaviours within the using system.

1.2 Issues outside the scope of this part of ISO/IEC 13522

The scope excludes any standardisation of models, services, systems, protocols or applications that are likely to make use of MHEG-5 objects.

The coded representation of content data is not in the scope of this part of ISO/IEC 13522.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 13522. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 13522 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of ISO and IEC maintain registers of currently valid International Standards.

— ISO/IEC 646:1991: *Information technology – ISO 7-bit coded character set for information interchange.*

- ISO/IEC 8824-1:1995 | ITU-T Recommendation X.680 (1994): *Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- ISO/IEC 8825-1:1995 | ITU-T Recommendation X.690 (1994): *Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.
- RFC 1521:1993: *MIME (Multipurpose Internet Mail Extensions) Part one: Mechanisms for specifying and describing the format of Internet message bodies*.

3 Terms and definitions

For the purposes of this part of ISO/IEC 13522, the following definitions apply:

3.1 abstract class

Class that is never instantiated into an interchangeable MHEG-5 object

NOTE - An abstract class defines attributes, behaviours and semantics of actions that may be exchanged as parts of any MHEG-5 object of the concrete subclasses of this abstract class.

3.2 action

Set of elementary actions

3.3 active

State of any MHEG-5 object when the Activation behaviour has been completed successfully for this object
An active object has its *RunningStatus* set to *True*.

3.4 application domain

Specific domain of applications making use of this part of ISO/IEC 13522 and providing additional tools and values in order to create a practical instance of an MHEG-5 environment

NOTE - More information on application domains is provided in Annex D of this part of ISO/IEC 13522.

3.5 application scope

Common scope of all MHEG-5 objects (scenes and ingredients) accessed from an MHEG-5 application

3.6 attribute

Named and typed value attached to a class

3.7 available

State of any MHEG-5 object when the Preparation behaviour has been completed successfully for this object
An available object has its *AvailabilityStatus* set to *True*.

3.8 base class

MHEG-5 class which defines some attributes, behaviours and semantics of actions that are shared by a given MHEG-5 class

3.9 concrete class

Class of any MHEG-5 object that may be encoded and interchanged according to the specifications provided in Annex A or Annex B of this part of ISO/IEC 13522

3.10 elementary action

Abstract representation of a message that may be sent to an object

NOTE - This part of ISO/IEC 13522 defines the semantics of available elementary actions for each MHEG-5 class. Note that the MHEG-5 class Action (with a capital A) has a different meaning described in clause 49 of this part of ISO/IEC 13522.

3.11 event

Abstract representation of an occurrence of a special meaning for any MHEG-5 object

NOTE - Events are used to trigger Link conditions and bring out the execution of sequences of elementary actions.

3.12 exchanged attribute

Attribute that is part of the interchangeable byte-code representation of an MHEG-5 object, and transmitted with that object

3.13 inactive

State of any MHEG-5 object when the Deactivation behaviour has been completed successfully or when no Activation behaviour has been applied successfully on this object
An inactive MHEG-5 object has its *RunningStatus* set to *False*.

3.14 inherited attribute

Attribute that is defined in a base class of the class of the MHEG-5 object

3.15 interchangeable representation

Octet string, which contains the encoded exchanged attributes of this MHEG-5 object complying with the ASN.1 syntax and encoding provided in Annex A of this part of ISO/IEC 13522, or, when textual representation is preferred, with Annex B of this part of ISO/IEC 13522

3.16 internal attribute

Abstract data structure, never byte-code encoded or exchanged, that is used to define semantics of internal behaviours or actions for any MHEG-5 object

NOTE - Any MHEG-5 engine might consider that an internal attribute is part of the internal representation of the MHEG-5 object; however, this is not mandatory. What is mandatory is to implement the functionality described by these internal attributes.

3.17 internal behaviour

Abstract function that defines the semantics of MHEG-5 elementary actions for any MHEG-5 class

NOTE - An internal behaviour of a class is most of the time overridden by internal behaviours of subclasses of this class. An MHEG-5 engine might consider that an internal behaviour of a class is a private method of this class, however this is not mandatory.

3.18 MHEG-5 application

Set of scenes and control information that allows the user to navigate between scenes

NOTE - the MHEG-5 class Application (with a capital A) has a different, more specific meaning that is given in clause 10 of this part of ISO/IEC 13522.

3.19 MHEG-5 class

Abstract definition of exchanged and internal attributes of parts of interchangeable multimedia / hypermedia objects as well as definition of the semantics of internal behaviours and the effect of MHEG-5 actions for these objects

3.20 MHEG-5 engine

Process or set of processes that interpret MHEG-5 objects encoded according to the encoding specifications defined in Annex A or in Annex B of this part of ISO/IEC 13522

3.21 MHEG-5 object

Instance of any MHEG-5 class

NOTE - An MHEG-5 object is not a physical object, but rather an abstraction that may have many representations of different types. Various software services handle such representations.

3.22 MHEG-5 scene

Structure that co-ordinates the presentation (visual and audible) of MHEG-5 objects

3.23 mix-in class

Abstract class that does not inherit from the Root class

Examples: *Interactable* class, *TokenManager* class.

3.24 non-available

State of any MHEG-5 object when the Destruction behaviour has been completed successfully or when no Preparation behaviour has been applied successfully on this object

A non-available object has its *AvailabilityStatus* set to *False*. Even if an MHEG-5 object does not exist in the MHEG-5 engine, its *AvailabilityStatus* exists and is set to *False*.

3.25 subclass

Any MHEG-5 class that shares the same attributes, behaviours and semantics of actions as another MHEG-5 class.

4 Conformance

This clause specifies conformance requirements for MHEG-5 engines and for MHEG-5 applications.

4.1 Conformance of MHEG-5 objects

Any MHEG-5 object shall have an octet representation. For interchange purposes, the octet representation shall be compliant with the ASN.1 syntax and encoding defined in Annex A, or with the textual notation grammar defined in Annex B. The application domain shall choose which representation to use: that of Annex A or that of Annex B; and that representation shall then be used exclusively throughout the application domain.

The attributes of any MHEG-5 object shall meet all requirements defined in the relevant subclauses of this part of ISO/IEC 13522.

4.2 Conformance of MHEG-5 engines

Conformance of MHEG-5 engines can only be measured with regard to a complete application domain definition. To fully specify conformance, an application domain shall define, in addition to the interchange representation, the following:

1. a set of classes from the list of all classes of this part of ISO/IEC 13522, as prescribed in 4.2.1
2. a set of features from the list in 4.2.2
3. additional concrete choices as listed in 4.2.3.

NOTE - Refer to Annex D of this part of ISO/IEC 13522 for an example of complete definition of an application domain.

4.2.1 Conformance to the acceptance of a set of Classes and Elementary Actions

Conformance to the acceptance of a set of Classes and Elementary Actions is defined as follows.

Any MHEG-5 engine is required to implement at least the following minimum set of classes:

- Application Class
All attributes, events and internal behaviours shall be implemented.
- Scene Class
All attributes, events and internal behaviours shall be implemented.
- Link Class
All attributes, events and internal behaviours shall be implemented.
- Action Class
All attributes, events and internal behaviours shall be implemented.

All application domains shall define compliance to a set of classes containing at least the minimum set above. An application domain may specify a larger set of classes and elementary actions for compliance; in any case, the application domain shall clearly list the classes and elementary actions supported.

When additional classes are implemented in any MHEG-5 engine, the engine shall implement all of their attributes, events, internal behaviours and elementary actions as defined in this part of ISO/IEC 13522, with the possible exception of optional features listed in 4.2.2. Concerning the Action class, the engine shall implement all effects of MHEG-5 elementary actions corresponding to the specified set of classes. It is the role of each application domain to choose and well define a set of classes that is required for that specific application domain.

If a class is not handled by an MHEG-5 engine, and an object of this class is sent to the MHEG-5 engine, this causes an error that is handled by the default error handling defined in subclause 53.7.

4.2.2 Conformance to a set of engine functionality

Conformance to a set of engine functionality is defined as follows.

Any MHEG-5 engine shall provide all normative mechanisms defined in clauses 51 to 54.

Any MHEG-5 engine shall implement all effects of MHEG-5 actions and the internal behaviours of MHEG-5 classes included in the definition of their application domain, except for the following optional features:

- Ancillary connections (corresponding to OpenConnection and CloseConnection actions),
- Caching, (corresponding to caching of MHEG-5 objects and content data of Ingredient objects),
- Cloning, (corresponding to the Clone action defined in Ingredient class).
- Free-moving cursor,
- Bitmap and Video scaling, (corresponding to the ScaleBitmap and ScaleVideo actions of the Bitmap and Video classes).
- Stacking of Applications, (corresponding to the Spawn action of the Application class),
- Trick mode, (corresponding to the SetSpeed action of the Stream class),

An application domain shall clearly define a list of which ones of the above features are mandatory or optional for conformance to the application domain.

4.2.3 Additional requirements for conformance specification

In addition to the two items above, the following tables shall be specified by a given application domain to fully define conformance.

NOTE - For each one of these table, a concrete example is given in Annex D (informative), thus defining an example of application domain.

- Content Data Encoding

The application domain shall specify which type of content data is supported and which type of encoding is supported. The following two tables shall be filled for each application domain.

Attribute	Permissible values
FontAttributes	
FontName	
AbsoluteColour	
CharacterSet	
TransitionEffect	

Encoding table:		
Type of content	Content Encoding	Hook Values (Integer)
Font encoding format		
Palette encoding format		
Bitmap encoding format		
Text encoding format		
EntryField encoding format		
HyperText encoding format		
Stream encoding format		
LineArt encoding format		
CursorShape encoding format		
InterchangedProgram encoding format		
AbsoluteColour encoding format		

- **UserInput registers**

In order to have a working instantiation standard, the application domain shall specify one or more InputEventRegisters. Each register has a number, which is exchanged as one of the parameters of a Scene object and a content (which is not exchanged) consisting of a set of numbers (representing UserInputEventTags) and a name. The name/number pairs bind a specific UserInputEventTag to a logical input event. It is the task of the engine implementer to bind the logical input event to one or more physical input events.

The following table shall be filled for each application domain.

Register #	UserInputEventTag	Semantics	Comment
(Integer)	...	...	...

- **EngineEvent**

An MHEG-5 application domain may specify a set of numbers associated with EngineEvent to distinguish between the various external events that lead to the generation of the EngineEvent. In such case, each of these specific EngineEvent shall be mapped to a corresponding Integer in table such as the one below. Values not reserved by the application domain are free for use by the application programmer.

EngineEvent	EventTag
...	(Integer)

- **GetEngineSupport**

Application domains may define, in addition to the strings mentioned in this part of ISO/IEC 13522, other permissible strings for the GetEngineSupport action. In such case, the application domain shall clearly list these additional strings.

- **Semantic constraints on the MHEG-5 applications**

For each feature defined by a string for the GetEngineSupport action, an application domain may choose to constrain its applications in some way. In such cases a table of constraints shall be provided.

Feature	Constraint
...	...

- **Protocol mapping and external interaction**

Finally there are the different actions that have an effect that is external to the runtime. These are functions that retrieve objects, manipulate streams and call external function. For interoperability, these actions shall have a common underlying external effect. This usually implies that there is a consistent mapping of these actions on the underlying external communications functions for users of MHEG-5 application domains. The following mappings shall be provided by the application domain:

MHEG-5 entity	Mapping needed.	Semantics of MHEG-5 structures that needs specification:
OpenConnection, CloseConnection	Mapping to connection management (and possibly session management) protocols in the application domain	• In OpenConnection: • Protocol • Address

RemoteProgram objects	Mapping to RemoteProgram call protocol in the application domain	• In Call and Fork: • Name • Parameters • ProgramConnectionTag
Application name space	Mapping to name space of the application domain	• ObjectReference • ContentReference
Application name space in case a TransitionTo action uses the ConnectionTag parameter	Mapping to the name space of the application domain	• ObjectReference • ContentReference
Persistent storage name space	Mapping to the name space of the persistent storage	• In StorePersistent and ReadPersistent: • InFileName, OutFileName
Stream actions	Mapping to the stream interface of the application domain	• In Stream • Speed • CounterPosition
Stream events	Mapping to stream states and stream events in the application domain	• In Stream • StreamPlaying, StreamStopped (mapping to application-domain stream state machine) • CounterPosition • StreamEventTag

5 Overview of the MHEG-5 Classes

This part of ISO/IEC 13522 was developed to support the distribution of interactive multimedia applications in a client/server architecture¹ across platforms of different types and brands. This part of ISO/IEC 13522 defines a final-form representation for application interchange. The applications consist mainly of declarative code, but provisions for calling procedural code have been made. MHEG-5 applications need only be authored once and then runs on any platform that is compliant to this part of ISO/IEC 13522. The developed applications would reside on the server, and as portions of the application are needed, they will be downloaded to the client. In a broadcast environment, this download mechanism could rely, for instance, on cyclic rebroadcasting of all portions of the application. It is the responsibility of the client to have a runtime that interprets the application parts, presents the application to the user, and handles the local interaction with the user.

Any MHEG-5 application is made up of scenes and objects that are common to all scenes. A scene contains a group of objects used to present information (graphics, sound, video, etc.) along with localised behaviour based on events firing (e.g. the Left button being pushed activates a sound). At most one scene is active at any one time. Navigation in an application is done by making transitions between scenes.

The interactive system has the ability to display visual objects in a rectangular coordinate system with a fixed size, and to play audible objects. User input devices (e.g. remote control, game controller, etc.) may be used with the runtime to allow interaction with the applications.

The figures in this informative clause present the class diagram of the object classes defined by this part of ISO/IEC 13522. Their meaning is explained in Figure 1.

	The boxes depict an MHEG-5 class. Class names in bold depict a concrete class.
	Class names in normal style depict an abstract class.
	Triangles depict inheritance relationships.
	Diamonds depict composition relationships.
	Dark circles depict a zero-to-more relationship.

Figure 1 - Legend of Class diagrams

The Figure 2 presents an overview of the top layer in the MHEG-5 class hierarchy. The next part of this clause introduces the concepts defined by this part of ISO/IEC 13522 by explaining the classes shown in this Figure and their subclasses.

¹ The server can be a *virtual* server, such as the collection of a number of broadcast channels on a broadcast network.

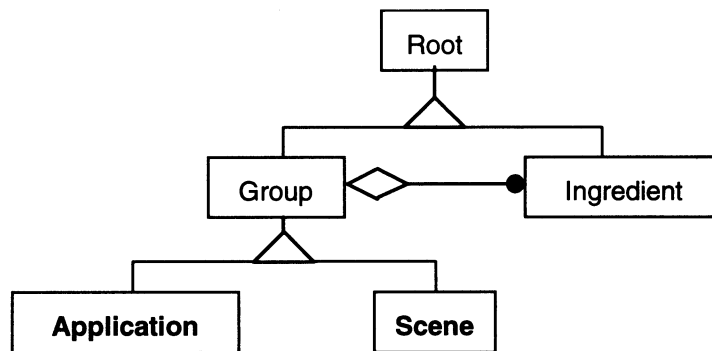


Figure 2 - Diagram of the top of the class hierarchy

5.1 Root

This is the abstract base class for all other MHEG-5 classes². Its main functionality is to provide semantics for generic MHEG-5 behaviour (activation, deactivation, preparation, destruction), and to provide a mechanism for object identification.

5.2 Group

This is an abstract base class for the classes Application and Scene. Its main functionality is that of allowing the grouping of objects of other classes for exchange between the MHEG-5 engine and other entities (similar to a «set» in standard object-oriented terminology). The objects that are grouped by this class are objects of the class Ingredient. Each Ingredient is always contained in exactly one Group. Objects within a Group may be referenced from objects in other Groups under certain conditions (see below).

5.3 Application

Objects of the Application class group objects of the Ingredient class. The Application class also has the semantic constraint that only one Application object may be active³ at once, and that no other objects may be active unless an Application object is active.

An idle MHEG-5 engine starts an application by preparing and activating the corresponding Application object. When the Application object becomes active, it automatically runs an OnStartup action, which can be used to run the first Scene object of the application. Since (exactly) one Application object is active whenever another object is active, the Ingredients contained in the Application object are visible and available to other objects that are active simultaneously. More specifically, any Ingredients contained in an Application object are available to the active Scene. This can be used to describe application-wide behaviour.

5.4 Scene

Objects of the Scene class group objects of the Ingredient class. The purpose of the Scene class is to allow spatially and/or temporally co-ordinated presentation. Only one Scene object may be active at a time within an MHEG-5 engine. A Scene object must be active in order for any ingredient (be it contained in a Scene or in an Application object) to be displayed.

The Scene class provides the special action TransitionTo, which makes it possible to perform a graphical transition between two scenes. Objects contained in a scene can only be displayed when that scene is active. Object that need to be displayed across several scenes (e.g., to have uninterrupted presentation over a scene transition must be contained in an application object. Finally, the Scene class provides information about the coordinate system to be used for visual presentation.

² With the exception of the abstract mix-in classes and of the Action class.

³ For the definition of the term active, see clause 3.

5.5 Ingredient

The Ingredient class is an abstract base class for the classes Link, Program, Palette, Font, CursorShape, Variable and Presentable. The subclasses of the Ingredient class are presented in Figure 3. The main functionality of the Ingredient class is to specify the generic behaviour of objects that can be part of a Scene or an Application object.

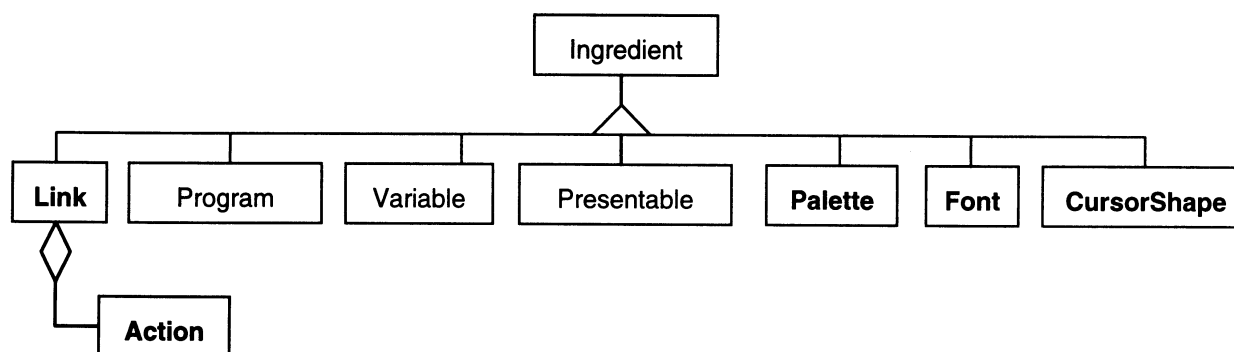


Figure 3 - Diagram of subclasses of the Ingredient class

5.6 Link

Link objects are used to express the behaviour of MHEG-5 applications. A Link object consists of a condition and an Action object. When the condition part evaluates to True, the Link is said to «fire»; this leads to the running of the associated Action object. The condition contains three parts: an event code (identifying which event has to be reacted on), a reference to the object from which the event should emanate, and a value that specifies the required value of the event parameter. In other words, a Link fires only if it is active and only if the right event is generated by the right object and contains the right event parameter. Active Links shall be part of either an active Scene or an active Application.

5.7 Action

An Action object has the functionality of executing, in synchronous sequence, a series of «elementary actions» as the result of a Link firing. An elementary action consists of the object to which the action is to be «targeted» and a list of values representing the parameters of the action. In fact, targeting an elementary action to an object corresponds to calling a method of an object in any ordinary object-oriented programming language. All elementary actions available are listed in clause 49.

The Action class does not inherit from any other MHEG-5 classes. Specifically, it does not inherit from Root, which means that Action objects cannot be addressed as individual entities.

5.8 Program

The Program class provides the functionality of calling a piece of procedural code from within the MHEG-5 context and exchanging parameters with it.

The Program class has the following three subclasses corresponding to the three types of procedural calls that can be made:

- ResidentProgram

Procedural call to a piece of code that is specific to the device on which the MHEG-5 engine is running. It can be used, for instance, to call device-specific runtime libraries.
- RemoteProgram

Procedural call to a piece of code that is located on a device different than the one where the MHEG-5 engine is running. It can be used, for instance, to implement a remote program call, where the actual body of the program is located at the server in a client-server system.

- **InterchangedProgram**

Procedural call to a piece of code which is exchanged as a part of an MHEG-5 object. The purpose of this class is to provide functionality needed for exchanging pieces of procedural code, and for calling those pieces of code.

5.9 Palette, Font, and CursorShape

The Palette class provides the possibility to encapsulate the encoded representation of a colour look-up table (CLUT). The function of a CLUT is to translate a colour index to a true colour value. A Palette may be used, for instance, with bitmaps to specify the colours in which the bitmap is to be rendered.

Similarly, the Font class allows applications to encapsulate the encoded representation of a font. A Font object, when associated with a Text object, is used to render the text of that object.

The CursorShape class, finally, allows applications to encapsulate the encoded representation of the bitmaps, mask, and other data needed to render a free-moving cursor. The free-moving cursor shape can be set and reset using a method of the Scene class.

For Fonts, Palettes and CursorShapes, the actual representation of the objects is not specified by this part of ISO/IEC 13522. However, an application domain where fonts and/or CLUTs and/or free-moving cursors are necessary features of applications can specify their encoding and semantics.

5.10 Variable

The Variable class provides the possibility to store and retrieve values. The Variable class has five subclasses corresponding to five different types of variables:

- BooleanVariable
- IntegerVariable
- OctetStringVariable
- ObjectRefVariable
- ContentRefVariable

Possible uses of Variables include parameter passing to and from Program calls, storage of the state of other MHEG-5 objects, passing indirect parameter values to actions and address indirection, i.e. as pointers to MHEG-5 objects.

5.11 Presentable

The Presentable class is an abstract base class for the MHEG-5 classes Audio, Visible, TokenGroup, and Stream. The subclasses of the Presentable class are presented in Figure 4.

Objects of the Presentable class represent information that can be directly seen or heard by the user. An important functionality of the Presentable class is to handle the actual encoded representation of the content data. This can be done either by «inclusion» or by «reference». In the former case, the content data is actually transmitted as part of the Presentable object itself; in the latter case, the Presentable object merely provides an external reference to the data.

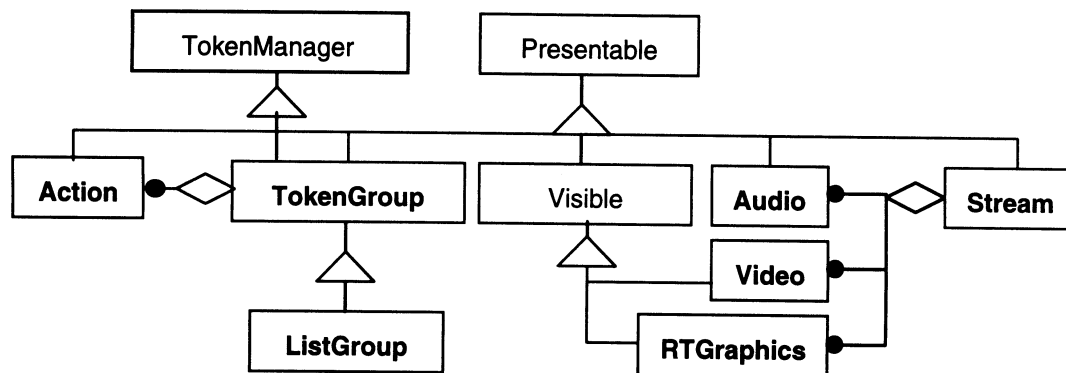


Figure 4 - Diagram of subclasses of the Presentable class

5.12 TokenGroup

The TokenGroup class provides the facility to navigate a logical token among a set of Visible objects. This structure can be used, for example, to manage the navigation of a «focus» among a set of buttons or other elements of a Scene. Some sets of actions may also be attached to the objects and executed on demand on the visible object that has the token. The latter feature provides a compact way of expressing behaviour when an element of the group gets or loses the focus.

5.13 ListGroup

The ListGroup class completes the TokenGroup class, providing functionality for selecting objects in a long list. It is best suited to implement a menu for selection, a group of checkboxes, a carousel of bitmaps, a fill-in form, a scrollable list of items, etc. In addition, the ListGroup class provides facilities to dynamically add and remove items to the group.

5.14 Stream

The Stream class defines a multiplex of continuous media for synchronisation in time. Audio, Video and RTGraphics objects might be elementary streams of a Stream multiplex: they are intended to be presented at the same time to the user. This structure can be used, for instance, to present video and audio synchronously and to switch from one audio channel to another. The content data of the Stream object is a reference to a real multiplex containing the elementary streams and some additional data for synchronisation. During the rendering process, the Stream player generates time-based events and marker-based events that might be used by the MHEG-5 application to trigger some Links.

5.15 Audio

The Audio class implements a sequence of audio data that can be used as an elementary stream of a Stream multiplex.

5.16 Interactable

The Interactable class is an abstract mix-in class inherited by the MHEG-5 classes HyperText, EntryField, Slider, and Button. Its main functionality is that of allowing the user to interact with objects of its sub-classes. These interactions enable the user to change the status and/or appearance of the objects, for instance by entering text in an EntryField object. When interaction is taking place, a certain class of events, called the «UserInput events» are not visible to Link objects, since those events are assumed to be used for the user interaction. The interaction can be aborted by targeting a certain action to the Interactable object.

Another functionality of the Interactable class is the ability to generate events associated with free-moving cursors (CursorEnter, CursorLeave).

5.17 Visible

The Visible class implements the functionality associated with rendering pieces of visual material at some location on the display screen. The subclasses of the Visible class are presented in Figure 5.

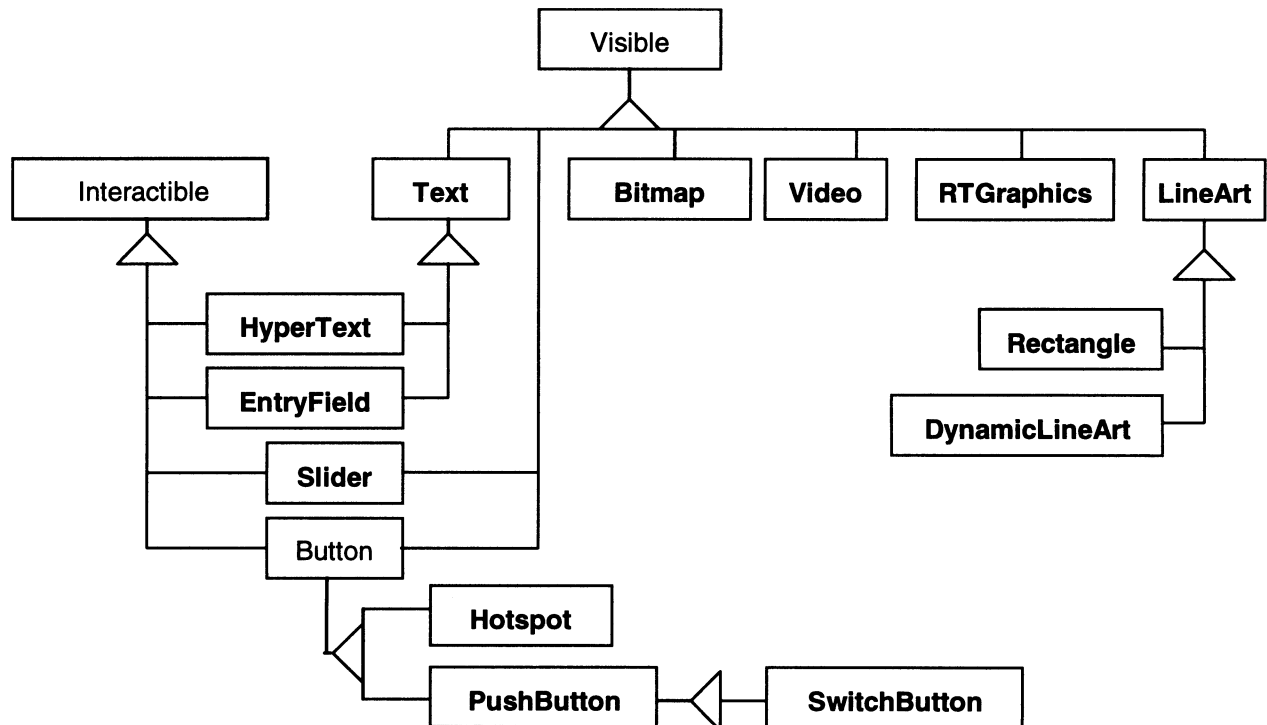


Figure 5 - Diagram of subclasses of the Visible class

The subclasses of the Visible class are described below:

- **LineArt**

An object of the LineArt class represents a graphical object with vectorial representation. It can be used, for instance, to present polyline objects, ellipses, bezier curves, etc.

- **DynamicLineArt**

An object of the DynamicLineArt class represents a graphical object that can be dynamically changed. It can be used to draw lines or curves that need to be presented on the fly.

- **Rectangle**

- **Bitmap**

- **Video**

- **RTGraphics**

An object of the RTGraphics (Real Time Graphics) class represents a stream of graphics objects that are displayed using autonomous placement and synchronisation. The RTGraphics stream can be used in conjunction with video and audio, for instance, to create a subtitling application.

- **Text**

Text objects represent text strings. A Text object may be associated with a Font object, which describes the font in which the text should be rendered (in case no font object is given, the default font is assumed). Text has two subclasses, HyperText and EntryField, which implement different types of interactive text.

- **Slider**

A Slider is an Interactable used to let the user set linearly a position within a certain range (given by a minimum and maximum value).

- Button

The Button class has the two subclasses: PushButton and Hotspot. The PushButton class has one subclass: SwitchButton. Buttons are rectangular areas on the screen with which the user can interact. Each of the three types of buttons has a specific event-generating behaviour attached to it. PushButtons and SwitchButtons are associated with a text item, which represents the text to be displayed in the middle of the button.

6 Structure of this part of ISO/IEC 13522

The following clauses of this part of ISO/IEC 13522 defines the semantics of the MHEG-5 classes. This is done on the basis of an abstract syntax notation used to describe the attributes of the exchanged objects. The semantics of the object classes are described in normative text. Clauses 51 to 54 of this part of ISO/IEC 13522 define some normative mechanisms that the MHEG-5 engine is required to implement. Annex A of this part of ISO/IEC 13522 defines the final-form syntax to be used for object exchange. Annex B of this part of ISO/IEC 13522, defines a textual interchange format which maps one-to-one on the final-form binary exchange format.

7 Notations

The following notations are used in the following clauses to describe the MHEG-5 classes defined by this part of ISO/IEC 13522.

<Name of the Class>

Description	<Short description of the semantics of the class>
Base class	<Name of the base class>
Subclasses	<List of subclasses, if any>
Status	<Abstract class Concrete class>

7.1 Attributes

This subclause defines inherited, exchanged and internal attributes for the class.

7.1.1 Inherited attributes

Requirements and constraints on attributes inherited from the base classes.

Attribute Name	Defined in	Constraints and Requirements
<Attribute Name>	<Class name>	<Specific constraints for the current subclass>.

7.1.2 Own exchanged attributes

List of exchanged attributes for this class.

<Attribute Name>	<Description of the attribute>
	<Type of the attribute>
	<Default value>

When the attribute is optional, default value is the value that shall be used when the attribute is not encoded. When the attribute is mandatory, default value is a hint on the most usual value to use.

7.1.3 Own internal attributes

List of internal attributes for this class.

<Attribute Name>	<Description of the attribute>
	<Initial value>

7.2 Events

These are the events that can be generated from objects of this class. They are used to express Link conditions that shall be checked when such an event is generated.

<Event Name>	<Description of the event>
	<Context of occurrence of the event>
	<Description of the event data associated with the event, if any>

7.3 Internal behaviours

The following internal behaviours are defined for most MHEG-5 classes:

- *Preparation*
- *Destruction*
- *Activation*
- *Deactivation*

In addition, the *Interaction* internal behaviour is defined for some MHEG-5 classes. These behaviours correspond to semantic requirements for operations internal to the MHEG-5 engine, similar to the notion of «private» methods of a class in object-oriented terminology. They are not encoded nor transmitted within an Application description.

When a behaviour is not described for an MHEG-5 class, the semantics of the behaviour for its base class applies.

<Behaviour name> <Semantics of the behaviour within the context of the current class.>

7.4 Effect of MHEG-5 actions

Define the semantics and syntax of MHEG-5 actions that may be targeted to the current MHEG-5 class.

<Action Name> <Semantics of the action within the context of the current class.>

7.5 Formal description

Description of an encoded MHEG-5 object of the current class in Extended Bacchus-Naur Form (EBNF):

Class Name	-->	First part, Second part
First part	-->	Subpart Alternative
Second part	-->	<i>Terminal (constant or type)</i>

8 Root Class

Description	Root class of all MHEG-5 classes.
Base class	None
Subclasses	Group, Ingredient
Status	Abstract class

8.1 Attributes

This subclause defines inherited, exchanged and internal attributes for this class.

8.1.1 Inherited attributes

This class has no inherited attributes.

8.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

ObjectIdentifier	This is a mandatory data structure, which consists of the following parts: - Optional GroupIdentifier. Unique identifier of a group of MHEG-5 objects. - Optional OctetString. - Default value: The GroupIdentifier of the Group within which the object is encoded. The actual structure of this parameter is not defined by this part of ISO/IEC 13522. However, the application domain shall define such a structure. See clause 51. - ObjectNumber. Unique identifier of any MHEG-5 object within a group. - Integer.
------------------	--

8.1.3 Own internal MHEG-5 attributes

This class defines the following internal attributes:

<i>AvailabilityStatus</i>	State of availability of the object. When the <i>AvailabilityStatus</i> of the object is True, the object is available, this means that the <i>Preparation</i> behaviour of the object has ended successfully. When the <i>AvailabilityStatus</i> of the object is False, the object is non-available, this means that the <i>Preparation</i> behaviour has not ended successfully or has not been invoked, or that the <i>Destruction</i> behaviour has been applied successfully. - Boolean value. - Default value: False.
<i>RunningStatus</i>	State of activity of the object. When the <i>RunningStatus</i> of the object is True, the object is active, this means that the <i>Activation</i> behaviour of the object has ended successfully.

When the *RunningStatus* of the object is False, the object is inactive, this means that the *Activation* behaviour has not successfully ended or has not been invoked, or that the *Deactivation* behaviour has been applied successfully.

- Boolean value.
- Default value: False.

8.2 Events

This class defines the following events:

IsAvailable This event is generated when the *AvailabilityStatus* attribute changes from False to True as a result of the *Preparation* behaviour ending successfully for the object.

NOTE - The purpose of this event is to indicate to the MHEG-5 engine that the object is available and can be activated.

- No associated data.

ContentAvailable This event is generated when the object and its content are available in an optimal state to the MHEG-5 engine. Its purpose is to indicate to the MHEG-5 engine that the activation behaviour can happen in a timely manner. This event is generated asynchronously with the *Preparation* behaviour for the object.

NOTE - Each MHEG-5 engine may choose to give a different meaning to that degree of content availability, this part of ISO/IEC 13522 does not specify any meaning or time requirement in this matter.

- No associated data.

IsDeleted This event is generated when the *AvailabilityStatus* attribute changes from True to False as a result of the *Destruction* behaviour ending successfully for the object.

- No associated data.

IsRunning This event is generated when the *RunningStatus* attribute changes from False to True as a result of the *Activation* behaviour ending successfully for the object.

- No associated data.

IsStopped This event is generated when the *RunningStatus* attribute changes from True to False as a result of the *Deactivation* behaviour ending successfully for the object.

- No associated data.

8.3 Internal behaviours

This class defines the following internal behaviours:

Preparation This behaviour has the basic semantics of allocating all requested resources in order to handle or to present this object.

Apply the following sequence of actions:

1. If the *AvailabilityStatus* attribute of the object is True, abort the behaviour. Otherwise:

2. Retrieve the object from an entity outside the engine.
 3. Set each internal attribute of the object to its initial value.
 4. Set the *AvailabilityStatus* attribute to True.
 5. Generate an *IsAvailable* event.
- The above steps are executed *synchronously*. The following step is asynchronous.
6. Generate a *ContentAvailable* event.

Destruction

This behaviour has the basic semantics of asking the MHEG-5 engine to delete the object.

Apply the following sequence of actions:

1. If the *AvailabilityStatus* attribute of the object is False, abort the behaviour. Otherwise:
2. If the *RunningStatus* attribute of the object is True,
 - a) apply the *Deactivation* behaviour.
 - b) wait for an *IsStopped* event from the object.

This shall all be done synchronously.
3. If the *RunningStatus* attribute of the object is False, execute the following actions synchronously.
4. If the *GroupCachePriority* attribute of the object itself or of the group this object belongs to is set to 0, the MHEG-5 engine shall free all resources allocated to the object.
Note that *GroupCachePriority* is defined in Group class.
5. If the *GroupCachePriority* attribute of the object itself or of the group this object belongs to is different from 0, the MHEG-5 engine may decide to either actually free all resources allocated to the object or to cache it.
6. Generate an *IsDeleted* event.

Note that the *IsDeleted* event shall be generated whether the resources mentioned above were actually freed or not; the object is deleted in the sense defined by this part of ISO/IEC 13522, even if some associated resources are not.

Activation

This behaviour has the basic semantics of immediately making this object active.

Apply the following sequence of actions:

1. If the *RunningStatus* attribute of the object is True, abort the behaviour. Otherwise:
2. If the *AvailabilityStatus* attribute of the object is False,
 - a) apply the *Preparation* behaviour to the object.
 - b) wait for an *IsAvailable* event from the object.

These steps are executed synchronously, meaning that the engine will not perform other actions until the *Preparation* behaviour has ended.

NOTES

1 The effect of the *Activation* behaviour (e.g., the display of a bitmap) will continue even after the behaviour itself has returned.

2 The generation of an *IsRunning* event and the modification of the *RunningStatus* internal attribute are parts of the *Activation* behaviour of subclasses of the Root class.

Deactivation This behaviour has the basic semantics of notifying the MHEG-5 engine to deactivate this object immediately.

Apply the following sequence of actions:

1. If the *RunningStatus* attribute of the object is False, abort the behaviour. Otherwise:
2. Set the *RunningStatus* attribute of the object to False.
3. Generate an *IsStopped* event.

8.4 Effect of MHEG-5 actions

This class defines the following applicable MHEG-5 actions:

GetAvailabilityStatus
(*AvailabilityStatusVar*) Set the Variable referenced by *AvailabilityStatusVar* to the value of the *AvailabilityStatus* attribute.

NOTE - A GetAvailabilityStatus action targeted to an object inexistant in the MHEG-5 engine is not an error; the result is False.

Provisions of use:

- *AvailabilityStatusVar* shall refer to an active BooleanVariable object.

Syntax description:

GetAvailabilityStatus	-->	Target, AvailabilityStatusVar
Target	-->	GenericObjectReference
AvailabilityStatusVar	-->	ObjectReference

GetRunningStatus
(*RunningStatusVar*) Set the Variable referenced by *RunningStatusVar* to the value of the *RunningStatus* attribute.

Provisions of use:

- The *Target* object shall be available.
- *RunningStatusVar* shall refer to an active BooleanVariable object.

Syntax description:

GetRunningStatus	-->	Target, RunningStatusVar
Target	-->	GenericObjectReference
RunningStatusVar	-->	ObjectReference

8.5 Formal description

Root Class	-->	ObjectIdentifier
ObjectIdentifier	-->	ObjectReference

9 Group Class

Description	Defines the structure and behaviour of objects used as composition of Ingredients.
Base class	Root
Subclasses	Scene, Application
Status	Abstract class

9.1 Attributes

This subclause defines inherited, exchanged and internal attributes for this class.

9.1.1 Inherited attributes

This class has all the attributes of its base class, with the following constraints:

Attribute Name	Defined in	Constraints and Requirements
ObjectIdentifier	Root	This attribute is mandatory for this class. The Group Identifier part of this attribute is mandatory and shall be unique within the name space of the application domain. The Object Number part of this attribute shall be set to zero.

9.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

StandardIdentifier	This is an optional sequence of two Integers. When encoded, - the first Integer shall always be set to 2, signifying «Joint ISO ITU». - the second Integer shall always be set to 19, signifying MHEG.
StandardVersion	This is a single optional Integer. It defines the version of this part of ISO/IEC 13522 to which the Group objects, and all its Items, conform. - Optional Integer. If encoded, its value shall be 1. - Default value: 1.
ObjectInformation	This is an optional OctetString. When encoded, it shall hold information about the objects encoded in this Group. Such information may comprise information about object Name, Owner, Version, Date, Keywords, Copyright, License, and Comments.
OnStartUp	Set of elementary actions to run during the <i>Activation</i> behaviour for the Group. - Optional inclusion of an Action object. - Default value: None.
OnCloseDown	Set of elementary actions to run at the beginning of the <i>Deactivation</i> behaviour for the Group. - Optional inclusion of an Action object. - Default value: None.

OriginalGroup- CachePriority	Hint to the MHEG-5 engine regarding the relevance of caching this Group of Ingredients when it is destroyed. Value of the <i>GroupCachePriority</i> when the Group is prepared. • Optional Integer within the range [0, 255]. • Default value: 127. • Specific value: 0 means cache is not allowed for this Group and its Ingredients. NOTE - As specified in 4.2.2, caching of any kind is an optional feature of an MHEG-5 engine.
Items	Set of Ingredient objects that belong to the Group. When the Group contains no Ingredients, this attribute shall not be encoded. When this attribute is encoded it shall contain at least one Ingredient. • Optional Attribute. • Sequence of inclusions of Ingredient objects. • Default value: None.

9.1.3 Own internal attributes

This class defines the following internal attributes:

<i>GroupCachePriority</i>	Hint to the MHEG-5 engine regarding the relevance of caching this Group of Ingredients when it is destroyed. The <i>GroupCachePriority</i> may be compared with the <i>GroupCachePriority</i> of other Group objects to determine which of a number of groups has the highest likelihood of being required again by the application, once it has been destroyed. A higher value indicates a higher level of priority. It is the responsibility of the application designer to keep these numbers to a consistent range. The MHEG-5 engine is recommended to cache objects with higher priority in preference to objects with lower priority. • Optional Integer within the range [0, 255]. • Initial value: Value of the OriginalGroupCachePriority attribute. • Specific value: 0 means cache is not allowed for this Group and its Ingredients.
---------------------------	---

9.2 Events

This class has the same events as its base class, with identical semantics.

9.3 Internal behaviours

The following internal behaviours semantics have changed from this object's base class:

<i>Preparation</i>	1. Apply the <i>Preparation</i> behaviour to all Ingredients of the Group that have the InitiallyActive attribute set to True and to all Programs of the Group that have the InitiallyAvailable attribute set to True in the order that they are listed in the Items attribute. 2. Apply the <i>Preparation</i> behaviour as inherited from the base class.
<i>Destruction</i>	1. Apply the <i>Destruction</i> behaviour to all Ingredients of the Group in the reverse order that they are listed in the Items attribute. 2. Apply the <i>Destruction</i> behaviour as inherited from the base class.

- Activation**
1. Apply the *Activation* behaviour as inherited from the base class.
 2. Run the action contained in the *OnStartUp* attribute.
 3. Apply the *Activation* behaviour to all *Ingredients* of the *Group* that have the *InitiallyActive* attribute set to *True*, in the order they are listed in the *Items* attribute.
 4. Set the *RunningStatus* attribute of the *Group* object to *True*.
 5. Generate an *IsRunning* event.
- Deactivation**
- If group is not active, ignore behaviour. If group is active, do the following three steps:
1. Run the action contained in the *OnCloseDown* attribute.
 2. Apply the *Deactivation* behaviour to all active *Ingredients* of the *Group*, in the reverse order they are listed in the *Items* attribute.
 3. Apply the *Deactivation* behaviour as inherited from the base class.

9.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with identical semantics. In addition, the following applicable MHEG-5 actions are defined:

SetCachePriority
(*NewCachePriority*) Set the *GroupCachePriority* attribute to *NewCachePriority*.

Provisions of use:

- The *Target* object shall be available.
- *NewCachePriority* shall be set within the range [0, 255].

Syntax description:

SetCachePriority	-->	Target, NewCachePriority
Target	-->	GenericObjectReference
NewCachePriority	-->	GenericInteger

9.5 Formal description

Group Class	-->	Root Class, StandardIdentifier?, StandardVersion?, ObjectInformation?, OnStartUp?, OnCloseDown?, OriginalGroupCachePriority?, Items?
StandardIdentifier	-->	Joint ISO ITU (2), MHEG (19)
StandardVersion	-->	INTEGER
ObjectInformation	-->	OctetString

OnStartUp	--> Action Class
OnCloseDown	--> Action Class
OriginalGroupCachePriority	--> <i>INTEGER</i>
Items	--> Item+
Item	--> ResidentProgram Class RemoteProgram Class InterchangedProgram Class Palette Class Font Class CursorShape Class BooleanVariable Class IntegerVariable Class OctetStringVariable Class ObjectRefVariable Class ContentRefVariable Class Link Class Stream Class Bitmap Class LineArt Class DynamicLineArt Class Rectangle Class Hotspot Class SwitchButton Class PushButton Class Text Class EntryField Class HyperText Class Slider Class TokenGroup Class ListGroup Class

10 Application Class

Description	Defines a set of Ingredient objects, which are shared within an application scope.
Base class	Group
Subclasses	None
Status	Concrete class

10.1 Attributes

This subclause defines inherited, exchanged and internal attributes for this class.

10.1.1 Inherited attributes

This class has all the attributes of its base class, with identical semantics.

10.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

OnSpawnCloseDown	Action object to be executed when the Application is closed by opening another Application via the Spawn action. It may be useful for instance in order to store information that will be used when restarting this Application. OnSpawnCloseDown may be invoked only by the Spawn action (see Spawn). • Optional inclusion of an Action object. • Default value: None.
OnRestart	Action object to be executed when the Application is restarted. This may be useful, for instance, in order to retrieve information that was stored by OnSpawnCloseDown. OnRestart may be invoked only by the Quit action (see Quit). • Optional inclusion of an Action object. • Default value: None.
DefaultAttributes	Defines default attributes Application-wide that shall be used as default values when a corresponding Ingredient attribute is not specified. The following attribute defaults can be set:
CharacterSet	Default character set, or set of character sets, for text rendering throughout the application, except when otherwise specified. This Integer shall be encoded with a value representing the set. The application domain shall define the range of CharacterSet and its semantics. NOTE - The <i>CharacterSet</i> attribute of Application provides the initial character set for all objects containing text in the Application that do not specify it. • Optional Integer. • Default value: None.
BackgroundColour	Default colour to be used to render the background of a text object. This attribute is interpreted as a zero-based index in the colour look-up table defined by the <i>PaletteRef</i> attribute, or as a direct colour value, depending on the attribute type.

- Optional Integer or OctetString. An Integer will be interpreted as an index in a Palette; an OctetString will be interpreted as a direct colour value.
- Default value: «transparent».

TextColour Default colour to be used to render the foreground of a text object. This attribute is interpreted as a zero-based index in the colour look-up table defined by the *PaletteRef* attribute, or as a direct colour value, depending on the attribute type.

- Optional Integer or OctetString. An Integer will be interpreted as an index in a Palette; an OctetString will be interpreted as a direct colour value.
- Default value: Any colour.

Font Default font to use when presenting a Text object.

The *Font* attribute represents either a name for a font (which is resident in the MHEG-5 engine) or a reference to a Font object.

When no font reference is encoded in Application, the Text object is presented using a default font of the MHEG-5 engine.

- Optional attribute.
- OctetString representing a FontName, or reference to a Font object.
- Default value: Default font.

FontAttributes Default Font attributes such as style, character size, text colour and background colour.

The exact encoding format of the *FontAttributes* attribute is related to the value of the type of Font object mentioned by the *Font* attribute.

- Optional OctetString.
- Default value: No specific attribute set.

BitmapContentHook Default hook value for all Bitmap objects.

- Optional Integer.
- Default value: None.

StreamContentHook Default hook value for all Stream objects.

- Optional Integer.
- Default value: None.

TextContentHook Default hook value for all Text objects.

- Optional Integer.
- Default value: None.

LineArtContentHook Default hook value for all LineArt objects.

- Optional Integer.
- Default value: None.

Interchanged-ProgramContentHook	Default hook value for all InterchangedProgram objects. • Optional Integer. • Default value: None.
ButtonRefColour	Default ButtonColour colour for Button rendering. • Optional Integer or OctetString. • Default value: None.
HighlightRefColour	Default HighlightRefColour for Interactibles. • Optional Integer or OctetString. • Default value: None.
SliderRefColour	Default SliderColour colour Slider rendering. • Optional Integer or OctetString. • Default value: None.

10.1.3 Own internal attributes

This class defines the following additional internal attributes:

<i>LockCount</i>	Specify whether or not the display screen is in a frozen state. When this attribute is positive and different from zero, the display shall not reflect any changes made to Visible objects that would normally result in a change in their rendering. However, all those changes shall be reflected at once as soon as the screen is unfrozen. The latter is signalled by the <i>LockCount</i> attribute being set to 0. Audio objects (of a Stream multiplex) will continue to play through a LockScreen action, i.e., they will continue to be heard. Visible objects that are part of a Stream multiplex shall continue to be played, but any other changes to these objects (e.g., position, volume, etc.) shall not be reflected until the screen is unfrozen; by continue to play is meant that they shall continue to move their <i>CounterPosition</i> attribute when the screen is locked, but, the updating of the graphical presentation of these objects during locked screen is optional. On some engines their physical rendering continues running, on some others the image is stopped until the screen is unlocked. When the screen unlocks, the rendering of the Visible objects shall be compliant with the value of the <i>CounterPosition</i> attribute. • Integer greater than or equal to zero. • Initial value: 0.
<i>DisplayStack</i>	Ordered list of references to Visible objects indicating how Visibles of the application are organised in graphics layers. Visibles at the bottom of the <i>DisplayStack</i> are displayed in the background of the screen and Visibles at the top of the <i>DisplayStack</i> are displayed in the foreground of the screen. Note that the <i>DisplayStack</i> may contain references to inactive Visible objects. In this case, inactive Visible objects simply do not appear on the screen but they remain as valid elements of the <i>DisplayStack</i>.

- Ordered list of ObjectReferences to Visible objects.
- Initial value: Empty list.

10.2 Events

This class has the same events as its base class, with identical semantics. In addition, the following events are defined:

- EngineEvent* This event is generated when a particular event has occurred in the environment of the MHEG-5 engine. This part of ISO/IEC 13522 does not specify any of these events; the application domain may specify the semantics of each of these events and the value of their associated data.
- Associated data: EventTag - Integer.

10.3 Internal behaviours

The following internal behaviours semantics have changed from this object's base class:

- Deactivation* Execute the following sequence of actions:
1. Apply the CloseConnection action to all opened auxiliary connections.
 2. Apply the *Deactivation* behaviour as inherited from the base class.

10.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with identical semantics. In addition, the following applicable MHEG-5 actions are defined:

- StorePersistent* (StoreSucceeded, InVariables, OutFileName) Requests the MHEG-5 engine to save data in such a way that it may later be retrieved by the ReadPersistent action.
- Data to be saved is passed through a set of Variables referenced by the *InVariables* parameter. These variables may contain Booleans, Integers, OctetStrings, ObjectReferences and ContentReferences.
- The data is saved in a file data structure. The *OutFileName* is another parameter of the StorePersistent action. This part of ISO/IEC 13522 does not define the nature, structure, ownership, protection or restriction of the file space. However, StorePersistent and ReadPersistent actions shall use the same file name space.
- The effect of the StorePersistent action is synchronous. On the successful completion of the StorePersistent action, the Variable referenced by *StoreSucceeded* shall be set to True, otherwise to False.

Example:

Consider the following elementary action included as part of the OnSpawnCloseDown attribute of the an Application,

```
:StorePersistent (("myApp" 0) ("myApp" 1) (("scene1" 1) ("scene1" 2)) "myfile.txt")
```

("myApp" 0) is the ObjectIdentifier of the current Application. ("scene1" 1) and ("scene1" 2) are ObjectIdentifiers of variables that hold information relative to the current Application, e.g. user information. These data will be stored in file named "myfile.txt". The variable ("myApp" 1) refers to a Boolean indicating whether the elementary action succeeded or not.

Data may be recovered at the beginning or restart of this Application (or of another one) to avoid asking the user for the same information twice.

Provisions of use:

- The *Target* object shall be the active Application object.
- *StoreSucceeded* shall refer to an active BooleanVariable object.
- *InVariables* shall be set to a non empty list of references to active Variable objects of any type.

Syntax description:

StorePersistent	-->	Target, StoreSucceeded, InVariables, OutFileName
Target	-->	GenericObjectReference
StoreSucceeded	-->	ObjectReference
InVariables	-->	ObjectReference+
OutFileName	-->	GenericOctetString

ReadPersistent
(*ReadSucceeded*,
OutVariables,
InFileName)

Request the MHEG-5 engine to read data that has been saved by the StorePersistent action.

Data to be read is recovered through a set of Variable objects, referenced by the parameter *OutVariables*.

The data has been stored in a file data structure. The *InFileName* is another parameter of the ReadPersistent action. This part of ISO/IEC 13522 does not define nature, structure, ownership, protection or restriction of the file space. However, StorePersistent and ReadPersistent actions shall use the same file name space.

The effect of the ReadPersistent action is synchronous. At successful completion of the ReadPersistent action, the Variable referenced by *ReadSucceeded* shall be set to True, otherwise to False.

Example:

Consider the following elementary action as part of the *OnRestart* attribute of the an Application,

```
:ReadPersistent (("myApp" 0) ("myApp" 1) (("scene1" 1) ("scene1" 2)) "myfile.txt")
```

The content of the variables indicated by ("scene1" 1) and ("scene1" 2) will be set to the values read from the file "myfile.txt" when the Application is launched.

Provisions of use:

- The *Target* object shall be the active Application object.
- *ReadSucceeded* shall refer to an active BooleanVariable object.
- *OutVariables* shall be set to a non empty list of references to active Variable objects.

Syntax description:

ReadPersistent	-->	Target,
----------------	-----	---------

		ReadSucceeded,
		OutVariables,
		InFileName
Target	-->	GenericObjectReference
ReadSucceeded,	-->	ObjectReference
OutVariables	-->	ObjectReference+
InFileName	-->	GenericOctetString

Launch Activate a new application by flushing the currently active one, if any.

Execute synchronously the following sequence of actions:

1. Apply the *Destruction* behaviour of the currently active Scene object, if any.
2. Apply the *Destruction* behaviour of the currently active Application object, if any.
3. Apply the *Activation* behaviour of the Application object to which the Launch action was targeted.

NOTE - Any events that are generated during the execution of these steps are queued and dealt with only after the entire sequence has ended.

Provisions of use:

- The *Target* object shall be a non-available Application object.

Syntax description:

Launch	-->	Target
Target	-->	GenericObjectReference

Spawn Activate a new application in such a way that the current application is restarted when the new application quits.

Execute synchronously the following sequence of actions:

1. Execute the *OnSpawnCloseDown* Action of the currently active Application object.
2. Store the *GroupIdentifier* of the currently active Application on the application identifier stack, if any.
3. Execute the effect of the Launch action as described above.

The application identifier stack is an optional feature of an MHEG-5 engine. If it is not implemented, or if the application identifier stack is full, this action shall be implemented as the Launch action.

Provisions of use:

- The *Target* object shall be an Application object that is not currently active while there is currently an active Application object.

Syntax description:

Spawn	-->	Target
Target	-->	GenericObjectReference

Quit Close an application and restart the previous application.

Execute synchronously the following sequence of actions:

1. Apply the *Destruction* behaviour of the currently active Scene object, if any.
2. Apply the *Destruction* behaviour of the target Application object.
3. If the MHEG-5 engine has not implemented an application identifier stack, or if the application identifier stack is empty, the MHEG-5 engine shall then return to an idle state. In all other cases, the following steps shall be performed:
4. Apply the *Activation* behaviour to the Application object whose Group Identifier is on the top of the application identifier stack. Note that this includes executing the *OnStartUp* Action of that object.
5. Remove the top entry from the application identifier stack.
6. Execute the *OnRestart* Action of the newly activated Application object.

Provisions of use:

- The *Target* object shall be the currently active Application object.

Syntax description:

Quit	-->	Target
Target	-->	GenericObjectReference

LockScreen Freeze the display screen and prevent from reflecting changes to Visible objects.

Execute synchronously the following sequence of actions:

1. Increment the internal attribute *LockCount* by 1.
2. If the *LockCount* attribute is now a strictly positive value, lock the display screen.

Provisions of use:

- The *Target* object shall be the active Application object.

Syntax description:

LockScreen	-->	Target
Target	-->	GenericObjectReference

UnlockScreen This action may refresh the display screen and reflect at once all changes to Visible objects.

Execute synchronously the following sequence of actions:

1. Decrement the internal attribute *LockCount* by 1. If the result is less than zero, set the *LockCount* attribute to 0.
2. If the *LockCount* attribute is equal to 0, refresh the display screen.

Provisions of use:

- The *Target* object shall be the active Application object.

Syntax description:

UnlockScreen	-->	Target
Target	-->	GenericObjectReference

OpenConnection
(OpenSucceeded,
Protocol, Address,
ConnectionTag)

Attempt to open a connection with an entity outside the MHEG-5 engine.

The OpenConnection action has the following parameters:

<i>OpenSucceeded</i>	If the OpenConnection action terminates successfully, the Variable referenced by <i>OpenSucceeded</i> shall be set to True, otherwise to False.
<i>Protocol</i>	Identifier of the protocol to be used when establishing the connection.
<i>Address</i>	Address of the counterpart with whom the connection should be made. The coding of this parameter depends on the value of the <i>Protocol</i> .
<i>ConnectionTag</i>	Integer used to reference the connection within the application.

Provisions of use:

- The *Target* object shall be the active Application object.
- *OpenSucceeded* shall refer to an active BooleanVariable object.

Syntax description:

OpenConnection	-->	Target, OpenSucceeded, Protocol, Address, ConnectionTag
Target	-->	GenericObjectReference
OpenSucceeded	-->	ObjectReference
Protocol	-->	GenericOctetString
Address	-->	GenericOctetString
ConnectionTag	-->	GenericInteger

CloseConnection
(ConnectionTag)

Attempt to close a connection with an entity outside the MHEG-5 engine.

The CloseConnection action has the following parameters:

<i>ConnectionTag</i>	Integer referencing a connection created by the OpenConnection action.
----------------------	--

Provisions of use:

- The *Target* object shall be the active Application object.
- If the connection referenced by *ConnectionTag* is not properly established, the CloseConnection action is ignored.

Syntax description:

CloseConnection	-->	Target, ConnectionTag
Target	-->	GenericObjectReference
ConnectionTag	-->	GenericInteger

GetEngineSupport
(*Feature, Answer*)

Return an Boolean that indicates if the MHEG-5 engine implements the specific option or set of options of this part of ISO/IEC 13522. The result of this action is returned in a BooleanVariable referenced by the *Answer* parameter and may be used to adapt the behaviour of the application to the engine's capacities.

Feature is a string encoded by ISO/IEC 646 and describing the option or set of options. Strings allowed are defined below; additional strings may be defined by the application domain. These strings are case sensitive, and integers are to be substituted to the symbols N, W, H, X or Y in parenthesis.

The answer to each of these strings shall be True or False.

AncillaryConnections

(asks whether engine supports ancillary point to point connections. These connections deal with the actions OpenConnection and CloseConnection, and the attribute ConnectionTag in various elementary actions)

ApplicationStacking

(asks whether engine provides support for the Spawn action of the Application class)

Cloning

(asks whether engine supports the Clone action)

FreeMovingCursor

(asks whether engine provides support for the class CursorShape, for the events CursorEnter and CursorLeave, and for the actions GetCursorPosition, SetCursorPosition, and SetCursorShape)

MultipleAudioStreams(N)

(asks whether engine supports at least N simultaneous Audio streams)

MultipleRTGraphicsStreams(N)

(asks whether engine supports at least N simultaneous RTGraphics streams)

MultipleVideoStreams(N)

(asks whether engine supports at least N simultaneous Video streams)

OverlappingVisibles(N)

(asks whether engine supports at least N overlapping Visibles)

Scaling

(asks whether engine supports ScaleBitmap and ScaleVideo actions)

SceneAspectRatio(W,H)

(asks whether engine supports a given aspect ratio. W & H are two integers, W / H is the width/height aspect ratio.)

SceneCoordinateSystem(X,Y)

(asks whether engine supports a given coordinate system. X & Y are two integers defining the coordinate system.)

TrickModes

(asks whether engine supports trick modes for Streams)

Provisions of use:

- The GetEngineSupport action shall be targeted only to the active Application object.
- *Answer* shall refer to an active BooleanVariable object.

Syntax description:

GetEngineSupport	-->	Target,
	-->	Feature,
		Answer
Target	-->	GenericObjectReference
Feature	-->	GenericOctetString
Answer	-->	ObjectReference

10.5 Formal description

Application Class	-->	Group Class, OnSpawnCloseDown?, OnRestart? DefaultAttributes?
OnSpawnCloseDown	-->	Action Class
OnRestart	-->	Action Class
DefaultAttributes	-->	DefaultAttribute+
DefaultAttribute	-->	CharacterSet BackgroundColour TextColour Font FontAttributes BitmapContentHook InterchangedProgramContentHook StreamContentHook TextContentHook LineArtContentHook ButtonRefColour HighlightRefColour SliderRefColour
CharacterSet	-->	INTEGER
BackgroundColour	-->	Colour
TextColour	-->	Colour
Font	-->	OctetString ObjectReference
FontAttributes	-->	OctetString
BitmapContentHook	-->	INTEGER
StreamContentHook	-->	INTEGER
TextContentHook	-->	INTEGER
LineArtContentHook	-->	INTEGER
InterchangedProgram- ContentHook	-->	INTEGER
ButtonRefColour	-->	Colour
HighlightRefColour	-->	Colour
SliderRefColour	-->	Colour

11 Scene Class

Description	Defines a set of Ingredient objects to be activated together.
Base class	Group
Subclasses	None
Status	Concrete class

11.1 Attributes

This subclause defines inherited, exchanged and internal attributes for this class.

11.1.1 Inherited attributes

This class has all the attributes of its base class, with identical semantics.

11.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

InputEventRegister	Register of permissible UserInput events for this Scene. While this Scene is active, the MHEG-5 engine shall generate only UserInput events that have an associated data consistent with the content of this InputEventRegister. The contents of InputEventRegisters is not defined by this part of ISO/IEC 13522. Integer identifying an InputEventRegister. Example: One could define two InputEventRegisters: one dedicated to moving pointer input events (MouseClicked, etc.) and another one dedicated to remote control input events (Up, Down, Left, Right, Enter, Quit, etc.). Knowing which type of input events are expected by the scene will allow the MHEG-5 engine to use the physical user devices to generate such events.
SceneCoordinate-System	Size of the coordinate system of this Scene. This attribute is expressed in numbers of rows and columns. Two Integers, expressing x-scene and y-scene sizes.
AspectRatio	Original aspect ratio of the Scene. This attribute is expressed by a width / height ratio. - Optional rational number. - Default value: 4 / 3.
MovingCursor	Indicates whether the Scene is expecting a free-moving cursor. Support by the MHEG-5 engine for free-moving cursors is optional. However, an application domain of MHEG-5 may declare such support to be mandatory. An engine that does not support free-moving cursors shall disregard this attribute. An engine that does support free-moving cursors shall act in the following way: - When this attribute is False, the engine shall not display a cursor on the screen. - When this attribute is True, the engine shall display a cursor on the screen. The

user shall be able to move this cursor to all positions within the Scene coordinate space. When the cursor enters (or leaves) the bounding box of an Interactable object, a CursorEnter (CursorLeave) event shall be generated for that Interactable. The engine shall support the SetCursorPosition and GetCursorPosition actions.

The following applies to this attribute in general:

- Optional Boolean.
- Default value: False.

NextScenes An optional list of OctetStrings, which shall be interpreted as GroupIdentifiers of Scene objects that might be presented after this one, along with a weight factor measuring the likelihood that these scenes are actually presented. The weight factor shall be an Integer in the range [0, 255] with 255 indicating the highest likelihood. It may be used by the MHEG-5 engine to resolve caching or pre-loading conflicts.

11.1.3 Own internal attributes

This class defines the following additional internal attributes:

Timers List of timers representing the temporal positions where the Scene shall receive *TimerFired* events.

Each timer has a unique identity number within the *Timers* list and a temporal position expressed in milliseconds. The temporal position is measured from the time origin of the timer. A timer shall be created by executing the SetTimer action on the active Scene; The time origin of the timer is by default the position in time where the SetTimer action, that created this timer, is executed; however, if the AbsoluteTime Boolean is encoded and set to True, the time origin of the timer is the position in time where the IsRunning event of the scene is generated.

- Sequence of the following data structures:
 - Timer identifier: Integer
 - Timer position: Integer
 - AbsoluteTime: Optional Boolean, default: False
- Initial value: Empty sequence.

11.2 Events

This class has the same events as its base class, with identical semantics. In addition, the following events are defined:

UserInput This event shall be generated by the MHEG-5 engine to indicate that user input has occurred.

- Associated data: UserInputEventTag - Integer. The value of the Associated Data shall be consistent with the content of the InputEventRegister attribute.

TimerFired This event is generated when a timer has fired.

- Associated data: TimerIdentifier - Integer.

11.3 Internal behaviours

This class has the same internal behaviours as its base class, with identical semantics.

11.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with identical semantics. In addition, the following applicable MHEG-5 actions are defined:

TransitionTo
(*ConnectionTag*,
TransitionEffect)

Check that target scene is different from active scene; if it is not, ignore action, if it is, remove the active Scene from the screen and replace it with the target Scene as follows.

Execute synchronously the following sequence of actions:

1. Apply the *Deactivation* behaviour to all active Ingredient objects of the currently active Application object that have the Shared parameter set to False in the reverse order that they are listed in the Items attribute of the application object.
2. Apply the *Deactivation* and *Destruction* behaviours to the Scene object currently active, if any. (This starts the transition effect.)
3. Apply the *Preparation* behaviour to the Scene object to which the TransitionTo action was targeted.
4. Apply the *Activation* behaviour to the Scene object to which the TransitionTo action was targeted. (This stops the transition effect.) The timeline for the new scene starts after generation of the *IsRunning* event.

This action has an optional parameter called *ConnectionTag*. If this parameter is not encoded, the Scene reference shall be resolved within the name space of the active Application object.

If the *ConnectionTag* parameter is encoded, the reference to the target Scene and all ContentReferences made from the target Scene shall be resolved within a name space that is used for communication over a communication link with the tag *ConnectionTag* (see the action OpenConnection in the Application class).

In addition, this action has a *TransitionEffect* parameter, which determines what type of visual transition effect to be implemented when performing the TransitionTo action. Implementing any transition effect is optional for the MHEG-5 engine. The encoding of the TransitionEffect attribute has to be specified by the application domain.

NOTE - for some transition effects the engine may be required to handle the destruction behaviour of Visibles differently from other deactivation (not introduced by TransitionTo). For example, the visual representation of the current scene may be saved in the graphical subsystem for a wipe or push effect, where the first scene is smoothly replaced by the second.

Provisions of use:

- The *Target* object shall be a non-available Scene object.

Syntax description:

TransitionTo	-->	Target, ConnectionTag?, TransitionEffect?
Target	-->	GenericObjectReference
ConnectionTag	-->	GenericInteger
TransitionEffect	-->	GenericInteger

SetTimer
(TimerId,
TimerValue,
AbsoluteTime)

Update the list of timers of the Scene.

Execute the following sequence of actions:

1. Update the *Timers* internal attribute of the Scene, according to the following rules:
 - a) If *TimerId* is the identifier of an existing Timer in the Scene, the new *TimerValue*, replaces the previous one. The parameter *AbsoluteTime* is ignored, in other words an absolute Timer cannot be replaced by a Timer relative to the Scene.
 - b) If there is no Timer with identifier *TimerId* in the Scene, insert a new Timer with identifier *TimerId* and values *TimerValue* and *AbsoluteTime* in the Scene. If *AbsoluteTime* is not encoded, it is set to False by default.
 - c) If *TimerValue* is not encoded and there is a Timer with identifier *TimerId* in the Scene, remove this Timer from the *Timers* list.
 - d) If *TimerValue* is not encoded and there is no Timer with identifier *TimerId* in the *Timers* list, discard this action.
2. The active Scene shall receive *TimerFired* events according to the new value of the *Timers* list.

If *AbsoluteTime* is set to True, the *TimerValue* parameter of this action shall be interpreted as a time offset from the time when the Scene is active. Otherwise, the *TimerValue* parameter of this action shall be interpreted as a time offset from the time when the action is invoked. In both cases, it is measured in milliseconds.

If the *TimerValue* parameter is zero and *AbsoluteTime* is False, the Timer shall be fired immediately.

Removing or changing a Timer does not suppress pending events from the former Timer.

Provisions of use:

- The *Target* object shall be the active Scene object.

Syntax description:

SetTimer	-->	Target, TimerId, TimerValue?, AbsoluteTime?
Target	-->	GenericObjectReference
TimerId	-->	GenericInteger
TimerValue	-->	GenericInteger
AbsoluteTime	-->	GenericBoolean

SendEvent
(EmulatedEvent-
Source,
EmulatedEvent-
Type,
EmulatedEventData)

Force the occurrence of an event.

Execute the following sequence of actions:

1. Generate an event corresponding to *EmulatedEventType*, *EmulatedEventSource* and *EmulatedEventData* as if it has been generated in the normal way.
2. Store this event in the synchronous or asynchronous event queue, according to *EmulatedEventType*.