INTERNATIONAL STANDARD

ISO/IEC 13522-6

First edition
1998-10-15

Information technology — Coding of multimedia and hypermedia information —

Part 6:
Support for enhanced interactive applications

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

Partie 6: Support pour les applications interactives améliorées

Reference number
ISO/IEC 13522-6:1998(E)

Contents

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 a vote.

International Standard ISO/IEC 13522-6 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*
- *Part 7: Interoperability and conformance testing for ISO/IEC 13522-5*

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

Information technology – Coding of multimedia and hypermedia information –

Part 6: Support for enhanced interactive applications

1 Scope

1.1 Context of the scope

ISO/IEC 13522 specifies the coded representation of multimedia/hypermedia information objects (MHEG objects) for interchange as final form units within or across services and applications, by any means of interchange including local area networks, wide area telecommunication or broadcast networks, storage media, etc.

MHEG objects can be produced by computer tools taking as source form multimedia applications designed using multimedia scripting languages. In this context, the MHEG script (or program) classes are intended to complement the other MHEG classes in expressing the functionality commonly supported by scripting languages. Script (or program) objects express more powerful control mechanisms and describe more complex relationships among MHEG objects than can be expressed by MHEG action and link objects alone. Furthermore, script (or program) objects express access to and interaction with external services provided by the run-time environment.

ISO/IEC 13522-5 defines the MHEG object classes for interchange and use in base-level applications intended to be run on limited resource terminals such as set-top-boxes in such contexts as interactive broadband services.

ISO/IEC 13522-5 defines the coded representation for program objects in an open manner so that program objects may encapsulate either standardised or proprietary program code. ISO/IEC 13522-5 allows program objects to include or reference programs that may be encoded in any encoding format as defined by the application domain.

1.2 Scope of this part of ISO/IEC 13522

The scope of this part of ISO/IEC 13522 is to define the semantics and final-form coded representation for the interchange of enhanced interactive multimedia applications.

These applications extend applications covered by ISO/IEC 13522-5 in incorporating functionality such as computing (data processing) and extended communication with the external environment, including servers, local devices, etc.

These applications may be exploited in any communication environment including broadcast-only mode, interactive client-server or peer-to-peer (conversational). However, the main focus is on interactive retrieval (client-server) applications running on limited resource set-top-units involving asymmetrical data interchange with real-time audiovisuals on the downstream channel.

The coded representation defined by this part of ISO/IEC 13522 specialises the coded representation defined by ISO/IEC 13522-5. Especially, this part of ISO/IEC 13522 defines the coded representation for the OriginalContent attribute of the MHEG-5 InterchangedProgram class. In addition, this part of ISO/IEC 13522

defines the Applet class; this subclass of InterchangedProgram features the ability to manage its own display and interaction, by delegation from the engine.

The resulting coded representation is

- compatible with that defined by ISO/IEC 13522-5;
- appropriate for execution on a set-top-unit with the same minimal resource constraints as expressed by ISO/IEC 13522-5.

This part of ISO/IEC 13522 specifies

- the interchange format for the OriginalContent attribute of the MHEG-5 InterchangedProgram class;
- the semantics of this coded representation;
- the coded representation and semantics of the Applet class;
- the semantic extensions to the MHEG-5 engine behaviour described by ISO/IEC 13522-5;
- the semantic restrictions on the MHEG-5 interchange format described by ISO/IEC 13522-5;
- the MHEG-5 API, which allows the code of an InterchangedProgram object to call upon the MHEG-5 engine's presentation functionality;
- the provisions for interworking between the MHEG-5 engine execution model and the execution model that underlies the program content interchange format.

MHEG engines are system or application components that handle, interpret and present MHEG objects. This part of ISO/IEC 13522 specifies the semantics of the MHEG-6 coded representation. These semantics are defined in terms of minimum requirements on the behaviour of MHEG-6 engines.

This part of ISO/IEC 13522 is applicable to all applications that interchange multimedia and hypermedia information.

2 Normative references

2.1 International standards

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, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

ISO/IEC 8825-1:1995, *Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.

ISO/IEC 10646-1:1993, *Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane*.

ISO/IEC 13522-5:1997, *Information technology - Coding of multimedia and hypermedia information - Part 5: Support for base-level interactive applications*.

2.2 Referenced specifications

All references in this subclause were correct at the time of approval of this part of ISO/IEC 13522. The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this part of ISO/IEC 13522. The reference to a specification within this part of ISO/IEC 13522 does not give it any further status within ISO/IEC; in particular, it does not give the referenced specification the status of an International Standard.

Lindholm, Tim and Yellin, Frank (September 1996), *The Java™ Virtual machine specification.* ISBN: 0-201-63452-X, Addison-Wesley Publishing Co.: Reading, Massachussets.

Gosling, James, Yellin, Frank and the Java team (May 1996), *The Java™ Application Programming Interface, Volume 1: Core Packages.* ISBN: 0-201-63453-8, Addison-Wesley Publishing Co.: Reading, Massachussets.

3 Terms and definitions

For the purposes of this part of ISO/IEC 13522, the terms and definitions given in ISO/IEC 13522-5 and the following terms and definitions apply.

3.1 applet
autonomous program that can be run only within a host framework

3.2 application class
JVM class entirely implemented in JVM code and interchanged as part of an MHEG-6 application

3.3 application programming interface (API)
boundary across which a software application uses facilities of programming languages to invoke software services

3.4 attribute
named, typed association between an object and a value, declared as part of the interface of a class:
a) MHEG-5 attribute (see ISO/IEC 13522-5);
b) attribute of a JVM class (see 2.2)

3.5 class
abstract definition of the data (attributes) and behaviours common to a set of interchanged information objects:
a) MHEG-5 class (see ISO/IEC 13522-5);
b) JVM class (see 2.2)

3.6 exception
signal that is raised when an exceptional condition occurs during the performance of the request to an operation; especially, JVM exception (see 2.2)

3.7 hypermedia, adj.
featuring access to monomedia and multimedia information by interaction with explicit links

3.8 instance
object that features the attributes and behaviours of a specified class

3.9 interface
description of a set of operations that a client may request of an object:
a) application programming interface;
b) JVM interface (see 2.2)

3.10 Java™[1] Virtual Machine (JVM)
the virtual machine defined by *The Java™ Virtual machine specification* (see 2.2), used as the interchange representation and execution model for the OriginalContent attribute of MHEG-6 InterchangedProgram objects

3.11 method
operation defined by a class; especially, JVM method (see 2.2)

3.12 MHEG-5 API
the API that defines the byte codes used by the OriginalContent of an MHEG-6 InterchangedProgram to access the attributes and control the behaviour of MHEG-5 objects

3.13 MHEG-5 InterchangedProgram object
MHEG-5 object that provides means to invoke a processing unit represented as interpreted or executable code consisting of sequences of instructions

3.14 MHEG-5 object
coded representation of an instance of an MHEG-5 class

3.15 MHEG-6, adj.
conforming to the provisions of this part of ISO/IEC 13522

3.16 MHEG-6 Applet object
instance of the Applet class defined in Clause 8

3.17 MHEG-6 application
application that involves the interchange, within itself or with another application, of MHEG-5 objects and of programs as the OriginalContent attribute of MHEG-5 InterchangedProgram objects, according to the representation defined by this part of ISO/IEC 13522

3.18 MHEG-6 engine
process or set of processes that can interpret MHEG-6 objects (including JVM programs) according to the provisions of this part of ISO/IEC 13522

3.19 MHEG-6 InterchangedProgram object
MHEG-5 InterchangedProgram object that conforms to the provisions of this part of ISO/IEC 13522

3.20 MHEG-6 object
MHEG-5 object that conforms to the semantic extensions defined by Clause 7, or object of the Applet class defined by Clause 8

3.21 MHEG-6 profile
profile of this part of ISO/IEC 13522

3.22 MHEG-6 program
list of JVM classes that are included or referenced by the OriginalContent attribute of an MHEG-6 InterchangedProgram object

[1] Java is a trademark owned by Sun Microsystems, Inc.

3.23 multimedia, adj.
that handles several types of representation media

3.24 multimedia and hypermedia application
application that features presentation of multimedia information to the user and interactive navigation across this information by the user

3.25 multimedia application
application that features presentation of multimedia information to the user

3.26 operation
service that can be requested and is provided by an object; it is defined within an interface by a name, a signature which defines the type of its parameters and return value, and the list of exceptions that its invocation may raise

3.27 program
sequence of binary codes that express computing behaviour and that can be run in an appropriate computer environment to effect this behaviour

3.28 Program content interchange format
the syntax and encoding for the OriginalContent attribute (when of the IncludedContent type) of an MHEG-6 InterchangedProgram, as defined in 12.1

3.29 scripting language
programming language intended for easy and rapid design of applications by non-professional programmers

3.30 stack
collection of elements that are inserted (pushed) and removed (popped) in last-in first-out (LIFO) order

3.31 system class
JVM class whose implementation is (at least partly) system-dependent (so consists of native code) and therefore must be available within the runtime environment for use by the VM

3.32 virtual machine (VM)
abstract specification of a micro-processor and its behaviour

NOTE A VM may be implemented on different hardware processors. A VM therefore implements the mechanism for all these processors to execute the same instruction set. It is also possible for a micro-processor to be designed so that its instruction set is identical to that of a VM. VM code can be used to make software portable.

4 Symbols and abbreviations

For the purposes of this part of ISO/IEC 13522, the following symbols and abbreviations apply.

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ETSI	European Telecommunications Standards Institute
IEC	International Electrotechnical Commission
ISO	International Organisation for Standardisation
ITU-T	International Telecommunication Union, Telecommunication standardisation sector
JVM	Java™ Virtual Machine
HTML	HyperText Mark-up Language
MHEG	Multimedia and Hypermedia information coding Experts Group
VM	Virtual Machine
WWW	WorldWide Web

5 Conformance requirements

This part of ISO/IEC 13522 defines conformance requirements

- on information objects, i.e. MHEG-6 objects;
- on implementations, i.e. MHEG-6 engine implementations.

5.1 Information object conformance

A conforming MHEG-6 object shall meet all of the following criteria:

- its encoding and syntax shall conform to the provisions referred to by 5.1.1;
- its semantics shall conform to the provisions referred to by 5.1.2.

The information object conformance is evaluated on the information objects that are interchanged for the purpose of their execution on a terminal.

5.1.1 Encoding and syntax

A conforming MHEG-6 object shall be encoded according to either the encoding rules and the syntax defined by Annex A, or those defined by Annex B.

Moreover, the attributes of a conforming MHEG-6 InterchangedProgram object shall follow the syntax and encoding provisions specified by 7.1.

5.1.2 Semantics

A conforming MHEG-6 object shall only include semantically valid constructs as defined by ISO/IEC 13522-5 and by Clauses 7 to 12 of this part of ISO/IEC 13522.

5.1.3 Profiles

This part of ISO/IEC 13522 defines no profiles.

5.2 Implementation conformance

An implementation of this part of ISO/IEC 13522 is an MHEG-6 engine.

This part of ISO/IEC 13522 defines the semantics of MHEG-6 objects. This implies conformance requirements not on information objects, but on the behaviour of MHEG-6 engines.

5.2.1 Conformance requirements

Conformance of MHEG-6 engines can only be measured with regard to an application domain definition, as defined by Clause 4 of ISO/IEC 13522-5.

In addition to all of the mandatory classes listed in Clause 4 of ISO/IEC 13522-5, any MHEG-6 engine shall interpret the following classes together with all of their attributes, events and internal behaviours:

- InterchangedProgram
- OctetStringVariable, IntegerVariable, BooleanVariable, ContentRefVariable, ObjectRefVariable

Any conforming MHEG-6 engine shall support the interpretation of any conforming MHEG-6 object whose class belongs to the application domain definition. Especially, a conforming MHEG-6 engine shall meet all of the following criteria:

- it shall conform to ISO/IEC 13522-5;
- it shall support the semantic provisions regarding MHEG-5 InterchangedProgram objects defined in Clause 7;
- if the Applet class is included in the application domain definition, then a conforming MHEG-6 engine shall support interpretation of MHEG-6 Applet objects, together with all of their attributes, events, internal behaviours and elementary actions, as defined in Clause 8;
- it shall support execution of JVM code as defined in Clause 9;
- it shall provide JVM code with full access to the kernel API defined in Clause 10;
- it shall provide JVM code with access to the MHEG-5 API defined in Clause 11 and Annex C, in either its reduced grade or its complete grade;
- it shall support the MHEG-5/JVM interworking provisions defined in Clause 12.

5.2.2 Conformance documentation

A conformance document with the following information shall be available for an implementation claiming conformance to this part of ISO/IEC 13522. The conformance document shall meet all of the following criteria:

- it shall list all the mandatory features required by this part of ISO/IEC 13522 or in ISO/IEC 13522-5, with reference to the appropriate Clauses and subclauses;
- it shall contain a statement that indicates the full names, numbers, and dates of the standards that apply;
- it shall state which of the optional features defined in this part of ISO/IEC 13522 or in ISO/IEC 13522-5 are supported by the implementation; for this purpose, it shall document all the application domain-dependent features as specified in Clause 4 of ISO/IEC 13522-5;
- it shall describe the behaviour of the implementation for all implementation-dependent features defined in this part of ISO/IEC 13522 or in ISO/IEC 13522-5. 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 part of ISO/IEC 13522 or ISO/IEC 13522-5 states that implementations may vary or where features are identified as undefined or unspecified.

5.3 Application conformance

Any InterchangedProgram object interchanged within a conforming MHEG-6 application (see 3.16) shall be a conforming MHEG-6 InterchangedProgram object. In addition, all objects of a conforming MHEG-6 application shall be encoded according to the same notation, either that defined by Annex A or that defined by Annex B.

6 Structure of this part of ISO/IEC 13522

The MHEG-6 specification consists of the following elements:

a) MHEG-5 objects as interchange units:

1) complying with the structure and semantics defined by ISO/IEC 13522-5;

2) specialised by syntax restrictions and semantic extensions on the InterchangedProgram class (see Clause 7);

3) extended by the new Applet class and the new Invoke action (see Clause 8);

4) with the coded representation defined by Annex A, that extends Annex A of ISO/IEC 13522-5 in a fully compatible way;

5) or with the coded representation defined by Annex B, that extends Annex B of ISO/IEC 13522-5 in a fully compatible way;

b) JVM code as the interchange format of the OriginalContent attribute of InterchangedProgram objects:

1) complying with the JVM coded representation and semantics (see Clause 9);

2) together with a kernel API (the java.lang, java.util and java.io packages) that provides JVM code with the required resident functionality (see Clause 10);

3) augmented by an MHEG-5 API (the iso.mheg5 package) that provides JVM code with access to MHEG-5 objects and control of MHEG-5 behaviour (see Clause 11 and Annex C);

4) completed by a set of provisions that express the interworking execution semantics, both regarding invocation of JVM methods from MHEG-5 objects and invocation of MHEG-5 elementary actions from JVM classes (see Clause 12).

7 MHEG-6 InterchangedProgram class

This Clause lists the semantic extensions and syntax restrictions applicable to the InterchangedProgram class and its subclasses, as well as the elementary actions that affect them.

7.1 InterchangedProgram object syntax

Unless otherwise specified in this subclause, any MHEG-6 object shall follow the MHEG-5 class syntax specified by ISO/IEC 13522-5.

This subclause specifies the restrictions on the syntax of MHEG-5 classes, i.e. the values, options or combinations with which MHEG-6 objects shall comply.

7.1.1 Syntax of InterchangedProgram class

Any MHEG-6 InterchangedProgram object is used to encapsulate one or several JVM classes, whose data are either included in the object or referenced by it.

7.1.1.1 Name attribute

The Name attribute of any MHEG-6 InterchangedProgram object shall be encoded as a sequence of null-terminated UTF-8 encoded strings. Each string shall represent the name of a JVM class encapsulated by the InterchangedProgram object.

NOTE The UTF-8 format for string encoding is defined by the JVM class file format referenced in 9.2. It features variable-length encoding of ISO/IEC 10646 (UCS) characters, so that all non-null ASCII characters are encoded using only one byte.

7.1.1.2 OriginalContent attribute

The OriginalContent attribute of any MHEG-6 InterchangedProgram object shall be encoded as follows:

a) if the OriginalContent attribute is of the IncludedContent type, its OctetString value shall follow the syntax defined by the Program content interchange format defined in 12.1.

b) if the OriginalContent attribute is of the ReferencedContent type, its ContentReference component shall consist of either of the following:

   1) a sequence of null-terminated ASCII encoded strings: each string shall represent the name of the file in which the data of the JVM class is stored. This sequence shall consist of the same number of strings as the Name attribute, and its file names shall be listed in the same order as the corresponding class names in the Name attribute; or

   2) a null string: in this case, the JVM class names (as expressed by the Name attribute) shall be mapped to local files using platform-dependent mapping rules.

7.1.1.3 ContentHook attribute

The ContentHook attribute of any MHEG-6 InterchangedProgram object shall be set to 0, the reserved value for programs encoded in JVM code.

7.1.1.4 Shared attribute

The Shared attribute of any MHEG-6 InterchangedProgram object that is interchanged as part of an Application object (i.e. has application-wide scope) shall be set to True.

7.1.2 Syntax of elementary actions applicable to InterchangedProgram objects

Any Call or Fork elementary action that is targeted at an MHEG-6 InterchangedProgram object shall have, within its Parameters argument, a first parameter Class of the GenericInteger type and a second parameter Method of the GenericOctetString type.

The Class parameter shall represent the index of a JVM class in the list of JVM classes defined by the Name attribute of the InterchangedProgram object, with 1 being the first index.

The Method parameter shall represent the UTF-8 encoded name of a static method of the class indicated by the Class parameter.

The Activation effect of the Call or Fork elementary action shall be to invoke the method indicated by Method on the class indicated by Class with the other parameters of the action passed (by value) as parameters to the method.

NOTE Any Parameter component of a Call or Fork elementary action that references an MHEG-5 variable (i.e. of the GenericObjectReference type) may be used by the program with input and/or output parameter semantics. If it is used as an output parameter, it need be set explicitly by the program using the SetVariable operation of the MHEG-5 API Variable class.

7.2 InterchangedProgram object semantics

This subclause describes in MHEG-5 terms the semantic extensions to the description of the MHEG-5 object behaviour.

7.2.1 InitiallyAvailable attribute

When the InitiallyAvailable attribute of an MHEG-6 InterchangedProgram object is set to True, the preparation behaviour of the InterchangedProgram object shall be triggered by the preparation of the MHEG-5 Group (Scene or Application) object to which the InterchangedProgram object is attached.

7.2.2 Scope of InterchangedProgram objects

As an instance of a subclass of the Ingredient class, any MHEG-5 InterchangedProgram object is attached to either a Scene or an Application object.

The scope of an MHEG-6 InterchangedProgram object is the MHEG-5 Group (i.e. Scene or Application) object to which it is attached. Any JVM class whose methods are invoked by a class of an MHEG-6 InterchangedProgram object shall be either of the following:

- a system class (see 3.31), provided by a standard package such as the kernel API;
- an application class (see 3.1), interchanged within an MHEG-6 InterchangedProgram object within the scope of the calling MHEG-6 InterchangedProgram object.

NOTE 1 In other terms, if a class of InterchangedProgram 1 calls a class of InterchangedProgram 2, then
- either InterchangedProgram 1 is part of a Scene, then InterchangedProgram 2 must be either part of the same Scene or part of the embedding Application;
- or InterchangedProgram 1 is part of an Application, then InterchangedProgram 2 must be part of the same Application.

A conforming MHEG-6 Application object shall not include any MHEG-6 InterchangedProgram object whose Java classes contain calls to methods of application classes that are out of scope.

NOTE 2 When non-conforming objects are provided to the engine, the behaviour of the engine is not specified.

Throughout a conforming MHEG-6 Application object, JVM classes included or referenced by InterchangedProgram objects shall all have distinct class names.

7.2.3 Effect of elementary actions applicable to InterchangedProgram objects

7.2.3.1 Preparation behaviour

When the Preload elementary action is targeted at any MHEG-6 InterchangedProgram object, the content of this InterchangedProgram object shall be retrieved.

The Java classes that this content encapsulates or references shall then be loaded into the VM by the MHEG-6 engine. This loading is performed asynchronously, i.e. may occur at any time before the activation behaviour starts.

If one such class has the same name as a class already loaded in the VM, the new class shall be ignored.

7.2.3.2 Activation behaviour

The Activation behaviour of an MHEG-6 InterchangedProgram object is triggered by a Call or Fork elementary action targeted at the object.

The main effect of the Activation behaviour shall be to invoke a method of a class, where

- the class is indicated by the first Parameter component of the Call or Fork elementary action, with the value of this parameter being the index of the class within the Name attribute of the InterchangedProgram object;

- the method is indicated by the second Parameter component of the Call or Fork elementary action, with the value of this parameter being the name of the method.

- the parameters of the method (if any) are indicated by the subsequent Parameter components of the Call or Fork elementary action, i.e. the first parameter of the method corresponds to the third Parameter component of the elementary action, and so on.

As part of the Activation behaviour, the result variable of the Call or Fork action (respectively the CallSucceeded or ForkSucceeded component) shall be automatically set to False and the action shall be terminated in all of the following cases:

- there are less than two Parameter components;

- the first parameter or the second parameter is not of a valid type as defined in 7.1.2;

- the first parameter does not refer to a valid index within the Name attribute (i.e. a value between 1 and the length of the list);

- the indicated JVM class has no method of the name indicated by the second parameter;

- the indicated method of the indicated JVM class is not a static method;

- the indicated method of the indicated JVM class has no signature to which the Parameters list could be resolved (using JVM matching rules), i.e. whose argument types match the Parameters' types (starting from the third component) one to one in the same order;

- the InterchangedProgram object is already active.

The MHEG-5 parameter types shall be mapped to JVM types as defined by Table 1.

Table 1 - Mapping of MHEG-5 parameter types to JVM types

MHEG-5 parameter type	JVM type
Boolean	Boolean
OctetString	OctetString
Integer	Integer
ContentReference	ContentReference
ObjectReference	ObjectReference

NOTE 1 The JVM Boolean and Integer types are classes defined by the kernel API (java.lang package). The JVM OctetString, ContentReference and ObjectReference types are classes defined by the MHEG-5 API (iso.mheg5 package) defined in Annex C.

EXAMPLE -
```
{ :interchgprg 100    :name "Prog" }
```
The elementary action call
```
:fork (100 99 :ginteger 1 :goctetstring "start"
       :gobjectref 150 :goctetstring :indirectref 2)
```
resolves statically to the signature
```
public static void Prog.start(ObjectReference,OctetString);
```

NOTE 2 If the Call or Fork elementary action has only two Parameter components, the signature of the method call should be
```
void <method_name>()
```
where <method_name> is replaced by the name of the method.

7.2.3.3 Deactivation behaviour

The Deactivation behaviour of an MHEG-6 InterchangedProgram object is triggered by either of the following:

- the effect of a Stop elementary action targeted at the object;
- the normal termination of all threads created from the method invoked by the Call or Fork elementary action that activated the object.

The Stop elementary action shall only apply to asynchronous programs, i.e. programs invoked using the Fork elementary action. Its effect shall be to terminate the program by stopping all VM threads created from it.

The normal termination of all threads created from the method invoked by a Fork elementary action causes the generation of an AsynchStopped event.

7.2.3.4 Destruction behaviour

The Destruction behaviour of an MHEG-6 InterchangedProgram may be triggered by an Unload elementary action or as a consequence of the destruction of the Group to which it is attached.

In any case, all the VM classes and objects it encapsulates shall be no longer accessible from the engine, unless they are also used by another InterchangedProgram.

Upon destruction of the MHEG-5 Application, all the VM objects and all the VM application classes (i.e. all but the system classes) shall be no longer accessible from the engine, and shall be dealt with by the engine so as to prevent any collision with classes of MHEG-6 applications to be interpreted later.

8 Applet class

Description	Defines encapsulation of JVM code with provisions for display and interaction capabilities
Base classes	InterchangedProgram, Interactible, Visible
Subclasses	None
Status	Concrete class

This Clause defines the Applet class using the notation defined by Clause 7 of ISO/IEC 13522-5.

Unlike the other MHEG-5 classes, the Applet class inherits from two subclasses of the Ingredient class. Whenever any conflict might occur due to diamond inheritance, the feature shall be inherited through the inheritance path defined by the first listed class. In other words, unless otherwise specified, all features of the Ingredient class (attributes, behaviours, events, effect of elementary actions) shall be inherited from the InterchangedProgram class, rather than from the Visible class.

8.1 Attributes

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

8.1.1 Inherited attributes

The Applet class has all the attributes of its base classes, with the following constraints:

Attribute name	Defined in	Constraints and requirements
InitiallyActive	Ingredient	As defined for Program class in ISO/IEC 13522-5, subclause 14.1.1
OriginalContent	Ingredient	As defined for MHEG-6 InterchangedProgram in 7.1.1.2 above and ISO/IEC 13522-5, subclause 17.1.1
ContentHook	Ingredient	As defined for MHEG-6 InterchangedProgram in 7.1.1.3 above and ISO/IEC 13522-5, subclause 17.1.1
Shared	Ingredient	As defined for MHEG-6 InterchangedProgram in 7.1.1.4 above
Name	Program	As defined for MHEG-6 InterchangedProgram in 7.1.1.1 above
InitiallyAvailable	Program	As defined for MHEG-6 InterchangedProgram in 7.2.1 above
OriginalPaletteRef	Visible	This attribute shall not be encoded for this class.

8.1.2 Own exchanged attributes

The Applet class defines no additional exchanged attribute.

8.1.3 Own internal MHEG-5 attributes

The Applet class defines no additional internal attribute.

8.2 Events

The Applet class has the same events as its base classes, with identical semantics except as specified below.

AsynchStopped This event shall be generated when the base JVM thread of the Applet object terminates.

- No associated data.

8.3 Internal behaviours

The Applet class has the same internal behaviours as its base classes, with the following constraints:

Preparation Apply the Preparation behaviour as defined for MHEG-6 InterchangedProgram in 7.2.3.1 above and ISO/IEC 13522-5, subclause 8.3, modified as follows:

1. Just before setting the AvailabilityStatus attribute to True, execute step 2 of the Preparation behaviour of the Visible class as defined in ISO/IEC 13522-5, subclause 31.3.

Destruction Apply the Destruction behaviour as defined for the MHEG-6 InterchangedProgram class in 7.2.3.4 above and ISO/IEC 13522-5, subclause 8.3.

Activation Execute the following sequence of actions synchronously:

1. Apply the Activation behaviour as defined for the Root class in ISO/IEC 13522-5, subclause 8.3.

2. Create a base JVM thread for the Applet object.

3. Invoke the constructor of the class whose index in the object's OriginalContent attribute is indicated by the first Parameter component of the Fork elementary action (as defined in 7.1.2). The JVM object created by this constructor shall be associated with the MHEG Applet object being activated. If the action has no parameter beyond the first, a constructor with no parameter shall be called. Otherwise, a constructor whose argument list types match the types of the Parameter components (starting from the second) of the Fork elementary action shall be called. The type matching rules defined in 7.2.3.2 shall be used.

   EXAMPLE - `{:Applet 200 :name "App" }`
   The elementary action call
   `:fork (200 99 :goctetstring "go")`
   resolves into calling the constructor:
   `public void App(OctetString)`
   with "go" as parameter.

4. If the constructor could not be invoked successfully, set the ForkSucceeded parameter to False and terminate the behaviour. This happens in all of the following cases:

   - the Fork elementary action has no parameter;

   - the first parameter of the Fork elementary action does not represent a valid index (from 1 to the length of the list) within the object's Name attribute;

- the indicated class has no constructor of an appropriate signature.

5. Display the Visible according to its position in the DisplayStack and to the position and the bounding box defined by the Position and BoxSize attributes (see also NOTE below).
6. Set the ForkSucceeded parameter to True.
7. Set the RunningStatus attribute to True.
8. Generate an IsRunning event.

Deactivation Execute the following sequence of actions synchronously:

1. If the RunningStatus attribute of the object is False, abort the behaviour.
2. If the InteractionStatus attribute of the object is True, interrupt the Interaction behaviour and set the InteractionStatus attribute to False.
3. Stop displaying the object.
4. Terminate the JVM base thread for the Applet object, as well as all threads created from it. Remove all references to the JVM Applet object from MHEG objects. The JVM object will be eligible for garbage collection when there are no longer any references to the object within the virtual machine.
5. Set the RunningStatus attribute to False.
6. Generate an IsStopped event.

Interaction This behaviour is identical to that defined for the Interactible class.

When the object's InteractionStatus attribute is True, all user input events are passed transparently by the engine to the applet. How these events are reported to the JVM code depends on the user interface API that may be defined by the application domain.

NOTE The Applet class may be used with a null box size to implement instantiable, non-visible programs, even in the absence of any underlying user interface API.

8.4 Effect of MHEG-5 elementary actions

The Applet class has the same set of MHEG-5 elementary actions as its base classes, with identical semantics, except as otherwise specified below.

SetData As defined for the Program class, this action shall not be targeted at an Applet object.

SetPaletteRef This action shall not be targeted at an Applet object.

Run This action shall not be targeted at an Applet object.

Call This action shall not be targeted at an Applet object.

Fork (*ForkSucceeded, Parameters*)

Apply the Activation behaviour.

Provisions of use:

- The ForkSucceeded parameter shall be set to an available BooleanVariable object.
- The first item of the Parameters list shall evaluate to a positive integer.
- Any further items of the Parameters list shall be set to values corresponding to the arguments of the constructor of the object's startup class. The order of items in the Parameters list shall correspond to the order of arguments of the constructor. All parameters shall be passed by value.

Stop

Apply the Deactivation behaviour.

In addition, the Applet class defines the following applicable MHEG-5 elementary action:

Invoke (*InvokeSucceeded, Method, Parameters*)

Request execution of a method on the applet associated with the object.

Execute the following sequence of actions synchronously:

1. If the Applet object is not active, disregard the action. Otherwise,
2. Invoke the method indicated by the Method parameter on the JVM object created by the Activation behaviour. If Parameters is an empty list, then the method shall be mapped to a method whose signature is
   `public void <method_name> ()`
   where `<method_name>` is replaced by the name of the method.
   Otherwise, a method whose argument list types match the types of the Parameter components (starting from the third) of the Invoke elementary action shall be called. The type matching rules defined in 7.2.3.2 shall be used.

   EXAMPLE - `{:Applet 200 :name "App" }`
   The elementary action call
   `:invoke (200 99 :goctetstring "start" :ginteger 1)`
   resolves into calling the method:
   `public void start(Integer)`
   with 1 as parameter.
3. If for any reason the method cannot be invoked successfully, set the InvokeSucceeded parameter to False and terminate the behaviour. This happens for instance when no method with the correct signature could be found.
4. Queue the method call in the applet's execution queue.

When all methods invoked by former Invoke actions have completed, execute the method call in the JVM thread created upon Activation.

NOTE The Invoke elementary action returns after step 4, i.e. once the method has been found and the method call has been queued.

Provisions of use:

- The InvokeSucceeded parameter shall be set to an available BooleanVariable object.
- The Method parameter shall evaluate to an octet string. It shall represent the UTF-8 encoded string of the name of a method.
- The items of the Parameters list shall be set to values corresponding to the arguments of the method. The order of items in the Parameters list shall correspond to the order of arguments of the method. All parameters shall be passed by value.

Syntax description:

```
Invoke              -->     Target,
                            InvokeSucceeded,
                            Method,
                            Parameters?
Target              -->     GenericObjectReference
InvokeSucceeded     -->     ObjectReference
Method              -->     GenericOctetString
Parameters          -->     Parameter+
Parameter           -->     GenericBoolean | GenericInteger |
                            GenericOctetString |
                            GenericObjectReference |
                            GenericContentReference
```

8.5 Formal description

```
Applet Class              -->   InterchangedProgram class,
                                Interactible class,
                                OriginalBoxSize,
                                OriginalPosition?
OriginalBoxSize           -->   XLength, Ylength
XLength                   -->   INTEGER
YLength                   -->   INTEGER
OriginalPosition          -->   XYPosition
```

9 Virtual machine

The virtual machine is the abstract specification of any computing entity that interprets VM code. This specification contributes to the definition of the semantics of MHEG-6 programs.

MHEG-6 programs shall have the coded representation and semantics defined by the JVM, as specified by Chapters 2 to 8 of *The Java™ Virtual machine specification* (see 2.2).

Only the parts of this referenced specification that specify requirements on the syntax and semantics of virtual machine code constitute normative provisions of this part of ISO/IEC 13522.

While Chapters 2 to 8 all contain such requirements, two chapters have a major importance in this specification: those that specify the instruction set and the class file format.

9.1 VM instruction set

The VM instruction set defines the set of virtual processor instructions that constitute the code part of VM code.

Any method of an MHEG-6 program shall use the VM instruction set specified by Chapter 6 "The Java Virtual Machine Instruction Set", pages 151 to 338 *of The Java™ Virtual machine specification* (see 2.2).

9.2 VM interchange format

The VM interchange format defines the syntax and encoding of VM code, organised into classes. A class file defines the structure of a class, as well as the data (constants, variables, attribute, values), and the code (sequences of instructions) that constitute this class.

Any class of an MHEG-6 program shall use the interchange format specified by chapter 4 "The `class` File Format", pages 83 to 138 of *The Java™ Virtual machine specification* (see 2.2).

NOTE The coded representation of an MHEG-6 program consists of a cluster of class files; it is defined in 12.1.

10 Kernel API

The kernel API consists of a set of basic functions necessary to the performance of JVM code for all MHEG-6 applications. Their implementation is at least partly system-dependent.

NOTE These functions especially include
- object and class concepts;
- wrapper classes for basic data types;
- character data handling;
- VM thread management;
- class loading;
- mathematical operations;
- exception and error handling;
- utility data structures;
- input and output streams.

10.1 Specification of the kernel API

The kernel API shall consist of the java.lang, java.util and java.io packages:

- the java.lang package is defined by the Chapter «Java Language Package», pages 1 to 185 of *The Java™ Application Programming Interface, Volume 1: Core Packages* (see 2.2);

- the java.io package is defined by the Chapter «Java I/O Package», pages 187 to 326 of *The Java™ Application Programming Interface, Volume 1: Core Packages* (see 2.2);

- the java.util package is defined by the Chapter «Java Utilities Package», pages 327 to 371 of *The Java™ Application Programming Interface, Volume 1: Core Packages* (see 2.2).

Only the parts of this referenced specification that specify requirements on the syntax and semantics of the above listed packages constitute normative provisions of this part of ISO/IEC 13522.

10.2 Syntax requirement

When calling upon the functions of the kernel API, MHEG-6 programs shall use the bytecodes implied by the package declaration of the kernel API.

Any MHEG-6 engine shall have an interface to all the classes, interfaces and exceptions according to the signatures defined by the kernel API.

10.3 Semantics requirement

Within an MHEG-6 engine, the implementation of the kernel API's classes, interfaces and exceptions shall behave as described by the kernel API specification referred to in 10.1. The kernel API shall be available to the MHEG-6 engine as soon as it starts and as long as it is running.

NOTE The trivial way is for the JVM kernel API to be resident within the run-time environment, but this is not required.

10.4 Pragmatics requirement

The pragmatics are specified by the application domain definition. In certain contexts, implementation of some functionality may be irrelevant. Profiles or application domains may therefore restrict the range of some behaviours or specify them as optional. They may also further define how described behaviours should be implemented in a given context.

11 MHEG-5 API

The MHEG-5 API consists of a set of interface functions used by JVM code to access the value of dynamic attributes of MHEG-5 objects and to invoke elementary actions on MHEG-5 objects.

11.1 Specification of the MHEG-5 API

The MHEG-5 API shall consist of the public interface of the iso.mheg5 package defined in Annex C. This interface is defined by the signature of the methods exposed by the classes and interfaces that constitute the package. The VM interchange format (see 9.2) defines an unambiguous way to encode these class declarations into an implementation of the package. The VM instruction set (see 9.1) defines an unambiguous way to encode invocation of the interfaces provided by such a package into VM code.

11.1.1 Design principles

The MHEG-5 API is designed according to the following principles:

- any MHEG-5 class is mapped to one Java class;
- any MHEG-5 elementary action is mapped to one Java method provided by the class that maps the MHEG-5 class to which the action is applied;
- any MHEG-5 dynamic attribute accessor is mapped to one Java "get" method provided by the class that maps the class holding this attribute;
- only those elementary actions that are used to invoke directly multimedia/hypermedia functionality (i.e. display, interaction and synchronisation) are provided as MHEG-5 API operations;
- failure to execute the invoked operations throws an MhegException, itself eventually extending Exception.

NOTE 1 The mapping of elementary actions to MHEG-5 API operations is provided in Annex D.

NOTE 2 Although they are mapped to elementary actions, some MHEG-5 API operations do not trigger elementary actions, but access directly the MHEG-5 internal data instead, as specified in 12.2.5.

11.1.2 Grades

Two grades are defined for the MHEG-5 API:

- the complete grade provides classes covering direct access to functionality of all MHEG-5 classes;
- the reduced grade is a subset of the complete grade, that provides only the following classes: ObjectReference, ContentReference, MhegException, OctetString, Root, Group, Application, Scene, Ingredient, Variable, BooleanVariable, IntegerVariable, OctetStringVariable, ObjectRefVariable, ContentRefVariable.

NOTE 1 In other words, only Variable, Group and their superclasses and subclasses, as well as the utility classes they refer to, are mandatory (this corresponds to the reduced grade). If the optional classes are selected, then it is mandatory that all of them be part of the package (this corresponds to the complete grade).

NOTE 2 The reduced grade allows MHEG-6 programs to access the same functionality of the complete grade, but in a less convenient way. Indeed, instead of calling directly the function that maps any given elementary action, a program using the reduced grade would use Variable.setVariable to set the MHEG-5 Variables to the intended parameters of that elementary action, then use Scene.sendEvent to trigger an MHEG-5 Link whose right part consists of that elementary action with reference to those variables as arguments.

11.2 Syntax requirement

Conformance is required at the level of the VM codes that express invocation of the MHEG-5 API operations. To invoke multimedia functionality, MHEG-6 programs shall use JVM instructions (invokevirtual, invokenonvirtual, invokestatic, invokeinterface) whose operands refer to method signatures (encoded as specified by the JVM class file format in 9.2) that are required to exactly match the signatures as specified by the MHEG-5 API (see Annex C).

NOTE This implies that applications can be designed in any language using any application programming interface to call upon MHEG-5 object functionality, provided that this language is then translated to VM bytecodes that are the same, as far as manipulating MHEG-5 objects is concerned, as the VM bytecodes that would be produced if the applications were designed in Java using the iso.mheg5 package. This principle also applies to the kernel API.

Any MHEG-6 engine shall have an interface to all the classes, interfaces and exceptions according to the signatures defined by the MHEG-5 API.

11.3 Semantics requirement

The MHEG-5 API, in either its reduced grade or its complete grade, shall be available to the MHEG-6 engine as soon as it starts and as long as it is running.

Within an MHEG-6 engine, the implementation of the MHEG-5 API's classes, interfaces and exceptions shall behave as described by the MHEG-5 API specification referred to in 11.1.

11.4 Pragmatics requirement

The pragmatics are specified by the application domain definition. As specified by ISO/IEC 13522-5, an application domain definition may restrict the range of some features or specify classes as optional. For instance, an MHEG-6 application domain definition may opt to require either the reduced grade or the complete grade. This has no impact on the syntax of the iso.mheg5 package provided. In any case, all the classes and methods defined by the chosen grade shall be provided. However, such a restriction impacts the behaviour of the iso.mheg5 package provided. Of course, the underlying behaviour of those classes and methods that map optional or unsupported features is not guaranteed. The exception code OPTION_NOT_SUPPORTED shall be returned whenever an MHEG-6 program calls upon a feature that is outside the application domain definition and not supported by the engine implementation.

11.5 Interworking considerations

All the constructors of any JVM class that map an MHEG-5 class (i.e. the Root class and all its subclasses) are made protected. This implies that Java references to MHEG-5 objects shall only be obtained by calling the static getObject method of the Root class. When an operation is invoked on this reference while it is no longer attached to an available MHEG-5 object, an exception shall be thrown.

The JVM «proxy» object mapping an MHEG-5 object should be created either when the MHEG-5 object's preparation behaviour is triggered or when the getObject operation is called for the first time for that MHEG-5 object. When the MHEG-5 object's destruction behaviour is triggered, this should be notified to the API support code that then makes the proxy object unavailable to the JVM application code.

NOTE 1 The proxy object need not be deleted explicitly due to the nature of the Java object model.

NOTE 2 The MHEG-5 API implies no assumption on the design of the MHEG-6 engine. For instance, the JVM and the MHEG-5 engine may be two separate processes that communicate through the MHEG-5 API. In another approach, the MHEG-5 API «proxy» objects may be the actual implementation objects used to represent the MHEG-5 objects in the MHEG-5 engine. The MHEG-5 API is only the way to ensure program code portability.

NOTE 3 If a reference is ambiguous because its groupIdentifier is not specified, it is resolved when getObject is called.

12 MHEG-5/JVM interworking provisions

12.1 Program content interchange format

Within an MHEG-6 InterchangedProgram object, any OriginalContent attribute of the IncludedContent type shall be encoded as defined by this subclause.

The formalism used is the same as for the JVM class file format defined in 9.2.

```
IncludedProgramContentData {
      u4[class_count] offsets;
      ClassFile[class_count] classes;
}
```

`offsets[i]` shall define the offset of the first byte of data of the i^{th} class (i.e. `classes[i]`) with regard to the start of the OriginalContent octetstring.

`class_count` shall have the same value as the number of character strings in the value of the Name attribute of the MHEG-6 InterchangedProgram object. This value defines the number of JVM classes encapsulated by the object.

`classes[i]` shall define the JVM class data, according to the class file format defined by 9.2.

Both `offsets` and `classes` arrays follow the class order defined by the Name attribute of the MHEG-6 InterchangedProgram object.

12.2 Semantics of elementary actions

12.2.1 Call

The Call elementary action has the semantics of a library function call. The effect of the Call elementary action is to run a specified method of the target class and to suspend the MHEG-5 engine operation until this method has exited and all the JVM threads created from it have exited as well.

An MHEG-6 InterchangedProgram activated using the Call elementary action shall only use the following MHEG-5 API operations:

- the operations that perform direct read access to the value of an MHEG-5 internal attribute (most of the methods whose name begins with "get");
- the setVariable methods provided by subclasses of the Variable class.

NOTE 1 The MHEG-5 API methods allowed to MHEG-6 InterchangedPrograms activated by Call are listed in Annex D.

Invocation of other MHEG-5 elementary actions from an InterchangedProgram activated using the Call elementary action is a programming error.

NOTE 2 To enforce this, every time an MHEG-5 elementary action is requested from VM code, the MHEG-5 API implementation should check that the VM thread of the calling VM code belongs to the VM thread group of an asynchronous program (i.e. invoked using Fork) and otherwise throw a fatal exception (see also 12.2.2).

NOTE 3 Hence (as specified in 12.2.5) the implementation of the allowed operations ("get" operations and setVariable) should access data through some direct access mechanism, rather than by invoking MHEG-5 elementary actions.

NOTE 4 When a Call elementary action is used, the MHEG-5 engine should not handle any elementary action until the call returns.

12.2.2 Fork

The Fork elementary action has the semantics of forking a new thread to run a program. The effect of the Fork elementary action is to provide an acknowledgement to the MHEG-5 engine, in the ForkSucceeded variable parameter, allowing it to resume operation.

Any thread created afterwards keeps on running while the MHEG-5 engine resumes processing.

An MHEG-6 InterchangedProgram or Applet activated using the Fork elementary action may invoke any MHEG-5 API operation.

NOTE 1 To enforce the difference between call and fork, the MHEG-5 API implementation should ensure that the Java thread created by a Fork elementary action, and any threads created from that, are in a distinct thread group from that which would be used when executing Java code in response to a Call elementary action.

An InterchangedProgram or Applet shall not create thread groups with arbitrary parents.

NOTE 2 This should be considered a programming error, resulting in an exception being thrown by the SecurityManager.

12.2.3 Invoke

The Invoke elementary action has the semantics of invoking a method on an applet. The effect of the Invoke action is to queue the method call, then to provide an acknowledgement to the MHEG-5 engine, in the InvokeSucceeded variable parameter, allowing it to resume operation. The method call is then executed in the Applet's main thread as soon as it is ready to receive it, i.e. when the previously queued method calls have been handled.

12.2.4 Stop

The effect of the Stop elementary action is to terminate any currently executing method and stop any VM thread activated by this program.

NOTE 1 The Stop elementary action only applies to programs activated using a Fork elementary action.

NOTE 2 To avoid any uncontrolled «crash» of the system, the application designer should make sure that all Java exceptions are handled explicitly by the program, so as to terminate the program in a clean fashion and return from the initial Call or Fork.

12.2.5 MHEG-5 API operations

All MHEG-5 API operations shall have synchronous execution semantics: whenever VM code invokes an MHEG-5 API operation, the VM thread in which this code is executed is suspended until the MHEG-5 API operation returns. Any MHEG-5 API operation that maps an elementary action shall return immediately after the requested MHEG-5 elementary action has been triggered and all its synchronous MHEG-5 consequences have been dealt with (e.g. execution of the elementary actions of the links triggered). Any MHEG-5 API operation used to access attributes shall return immediately after the requested attribute has been retrieved.

The operations that perform direct read access to the value of an MHEG-5 internal attribute (i.e. the "get" operations) and the setVariable operation shall not trigger MHEG-5 elementary actions, but instead bypass the action execution mechanism and access attributes or modify variables directly.

12.3 Execution semantics

This subclause describes from a JVM viewpoint the effect of the actions applicable to MHEG-6 InterchangedProgram objects.

For this purpose, it introduces the ClassMapper concept to illustrate and clarify the mechanisms specified in 7.2.3, and make recommendations on their implementation.

NOTE This description is intended for use as a reference of how an engine is expected to behave. There is no requirement to implement an MHEG-6 engine as described in this subclause.

The ClassMapper is a virtual entity used to describe how the association between an MHEG-6 InterchangedProgram object and a JVM class is dynamically maintained throughout the life of both the MHEG-6 object (which remains under the control of the MHEG-5 engine) and the JVM application class (once it becomes known to the VM).

12.3.1 Engine bootstrapping

The MHEG-6 engine is assumed to be "loaded" before any MHEG-6 application can be prepared. This means that at all times during interpretation of an application,

- the MHEG-5 engine is ready to process active Links;
- the ClassMapper is ready to receive any request;
- the JVM is ready to execute VM code.

12.3.2 ClassMapper initialisation

Activation of an Application object notifies the ClassMapper of the GroupIdentifier that a new application is starting.

NOTE 1 For this purpose, the ClassMapper could provide a function such as
```
public void setApplication (MHEGId application_id)
```

As soon as a Group object is prepared, descriptive information of all MHEG-6 InterchangedProgram objects attached to this Group is provided to the ClassMapper. For each MHEG-6 InterchangedProgram object, this descriptive information includes the names of the classes (as provided by the Name attributes), and references to their content data (through either class data offsets - for IncludedContent attributes - and/or class file names - for ReferencedContent attributes). For each of the classes, the ClassMapper can exploit the descriptive information to add an entry to its internal class map, which maps a class name to the encapsulating MHEG-6 InterchangedProgram object and a reference to the class data.

The use of this notification is to make it known to the ClassMapper that these classes can be loaded whenever required for resolution.

NOTE 2 The ClassLoader class of the java.lang package is an abstract class. An MHEG-6 ClassLoader (inheriting from the java.lang ClassLoader) need be implemented according to the MHEG-6 context. Parsing the program code interchange format (for included content data) and accessing the class files (for referenced content data) is performed by the loadClass method of the ClassLoader class through the data input mechanism (such as java.io.DataInput) available on the run-time environment (this data input mechanism is not specified by this part of ISO/IEC 13522).

EXAMPLE Assume the application contains an MHEG-6 InterchangedProgram object with class Add (used throughout the application). The name and location is notified to the ClassMapper, but the class is not loaded into the VM. Now assume scene 1 contains an InterchangedProgram with class Fact (used only within the scene), whose InitiallyAvailable attribute is set to True. So Class Fact has to be loaded into the VM. As a method of class Fact calls upon a method of class Add, loading of class Fact cannot be performed until the reference to class Add is resolved. The map maintained by the ClassMapper is therefore used to load the code of class Add, without disturbing the operation of the MHEG-5 engine.

NOTE 3 For this purpose, the ClassMapper could provide a function such as
```
public void notify (MHEGId object_id, String[] class_names,
                    ByteStreamReference[] class_data)
```
The structure of the map maintained by the ClassMapper could be as follows:
```
    ClassEntry[] class_map;
    ClassEntry {
```

```
        CONSTANT_Utf8_info name;
        MHEGId program_id;
        ByteStreamReference location;
    }
```

with ByteStreamReference being an implementation-dependent type representing a handle to large data areas used to hold referenced and/or included content.

12.3.3 Program preparation

Upon preparation of the MHEG-6 InterchangedProgram object, its OriginalContent data are retrieved and provided to the ClassMapper. The ClassMapper then parses the class file format and loads the classes into the VM.

NOTE For this purpose, the ClassMapper could provide a function such as
`public Error load (String class_name)`

12.3.4 Program activation

Upon triggering a Call or Fork elementary action on an InterchangedProgram, the MHEG-5 engine handles the first parameter (class index) and uses the Name attribute to retrieve the name of the target class, then the ClassMapper

- checks that the invoked class has a method of the name given by the second parameter (method);
- transforms the other MHEG-5 parameters into JVM Object parameters;
- if the method cannot be invoked (see 7.2.3), returns False (this is then used by the MHEG-5 engine to set the CallSucceeded or ForkSucceeded action parameter);
- actually calls the requested method of the JVM class with the transformed parameters;
- in the case of a Call, returns an exit code upon termination of the requested method (this is used by the MHEG-5 engine to set the CallSucceeded action parameter);
- in the case of a Fork, returns the identifier of the newly created ThreadGroup back to the MHEG-5 engine which temporarily stores it for further reference (see 12.3.5); then, upon termination of the requested method, sends the MHEG-5 engine a notification with an exit code used to set the ForkSucceeded action parameter.

NOTE For this purpose, the ClassMapper could provide functions such as

```
public boolean call (String class_name, String method_name, Param[] args)
public int fork (String class_name, String method_name, Param[] args)
```

12.3.5 Program deactivation

Upon triggering a Stop elementary action on an InterchangedProgram, the MHEG-5 engine uses the temporarily stored thread group identifier to request termination of the program. The effect of the Stop elementary action is to terminate all threads created by the Program, including the initial thread.

NOTE For this purpose, the ClassMapper could provide a function such as
`public void stop (int ThreadGroupId)`

12.3.6 Program destruction

Upon destruction of the MHEG-6 InterchangedProgram object, classes carried by this InterchangedProgram, whose names are defined by its Name attribute, are marked for deletion within the VM.

NOTE 1 For this purpose, the ClassMapper could provide a function such as
`public void unload (String class_name)`
This also removes the entry in the ClassMap.

Upon destruction of the MHEG-5 Application object, the ClassMapper is restored to its initial state.

NOTE 2 Although the JVM classes and objects are marked for deletion when their encapsulating InterchangedProgram is destroyed, there is no way of knowing when these objects have actually been deleted, as this depends on the garbage collection policy of the JVM implementation.

NOTE 3 If a JVM object is referenced by a class variable, then that object (and any object it refers to) remains until the class is unloaded.

NOTE 4 If an object of a class with Scene scope is referenced by an object of a class with Application scope, the code of this class may need to remain in the VM method area after the destruction of the Scene. However, such programming is deprecated.
More generally, if a class of a destroyed InterchangedProgram is used by another InterchangedProgram that is still available, the code of this class may need to remain in the VM method area until it is no longer referenced.

12.3.7 ClassMapper for Applet

The ClassMapper has a similar behaviour for MHEG-6 Applet objects. In addition,

- it maps the applet's MHEG reference to the corresponding Java instance and thread objects; after the activation behaviour calls upon the constructor of the target class, both of these have to be stored throughout the lifetime of the MHEG applet object;

- it translates the Invoke elementary action into method calls to be queued.

Annex A
(normative)

ASN.1 notation

This Annex describes the ASN.1 notation for the syntax of MHEG-6 objects.

Use of the ASN.1 notation is recommended for service delivery and deployment. Indeed the ASN.1 notation is considered to generally result in binary form three or four times more compact than the text notation.

Any MHEG-6 object encoded using the ASN.1 notation shall be described using the following ASN.1 syntax and shall be encoded according to the Distinguished Encoding Rules (DER) defined in ISO/IEC 8825-1:1995 | ITU-T Recommendation X.690.

The syntax below is compatible with that defined by Annex A of ISO/IEC 13522-5. So any MHEG-5 object encoded according to Annex A of ISO/IEC 13522-5 is also an MHEG-6 object, i.e. conforms to the syntax in this Annex.


```
--Copyright statement:
--© International Organisation for Standardisation 1998
--Permission to copy in any form is granted for use with conforming MHEG-6 engines and applications
--provided this notice is included in all copies

--$PREFIX=ISOMHEG-mheg-6:mheg-6
-- Module: mheg-6

ISO13522-MHEG-6 {joint-iso-itu-t(2) mheg(19) version(1) mheg-6(18)}
DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- This module defines the MHEG-6 abstract syntax which consists of data values of type:
-- ISO13522-MHEG-6.InterchangedObject.
-- This abstract syntax is identified by the name:
-- {joint-iso-itu-t(2) mheg(19) version(1) mheg-6(18)}.

InterchangedObject ::= CHOICE {
  application [0] ApplicationClass,
  scene       [1] SceneClass
}
```

-- <A.1> Root Class

```
RootClass ::= ObjectReference
```

-- <A.2> Group Class

```
GroupClass ::= SET {
  RootClass (WITH COMPONENTS
    {external-reference (WITH COMPONENTS {..., object-number (0)}) PRESENT}),
  standard-identifier           [2] StandardIdentifier OPTIONAL,
  standard-version              [3] INTEGER (1) OPTIONAL,
  object-information            [4] OCTET STRING OPTIONAL,
  on-start-up                   [5] ActionClass OPTIONAL,
  on-close-down                 [6] ActionClass OPTIONAL,
  original-group-cache-priority [7] INTEGER (0..255) DEFAULT 127,
  items                         [8] SEQUENCE SIZE (1..MAX) OF GroupItem OPTIONAL
}

StandardIdentifier ::= SEQUENCE {
  joint-iso-itu INTEGER (2),
  mheg          INTEGER (19)
}

GroupItem ::= CHOICE {
  resident-program      [9] ResidentProgramClass,
  remote-program        [10] RemoteProgramClass,
  interchanged-program  [11] InterchangedProgramClass,
  palette               [12] PaletteClass,
  font                  [13] FontClass,
```

```
  cursor-shape          [14] CursorShapeClass,
  boolean-variable      [15] BooleanVariableClass,
  integer-variable      [16] IntegerVariableClass,
  octet-string-variable [17] OctetStringVariableClass,
  object-ref-variable   [18] ObjectRefVariableClass,
  content-ref-variable  [19] ContentRefVariableClass,
  link                  [20] LinkClass,
  stream                [21] StreamClass,
  bitmap                [22] BitmapClass,
  line-art              [23] LineArtClass,
  dynamic-line-art      [24] DynamicLineArtClass,
  rectangle             [25] RectangleClass,
  hotspot               [26] HotspotClass,
  switch-button         [27] SwitchButtonClass,
  push-button           [28] PushButtonClass,
  text                  [29] TextClass,
  entry-field           [30] EntryFieldClass,
  hyper-text            [31] HyperTextClass,
  slider                [32] SliderClass,
  token-group           [33] TokenGroupClass,
  list-group            [34] ListGroupClass
  applet                [237] AppletClass
}
```

-- <A.3> Application Class

```
ApplicationClass ::= SET {
  COMPONENTS OF GroupClass,
  on-spawn-close-down [35] ActionClass OPTIONAL,
  on-restart          [36] ActionClass OPTIONAL,
  default-attributes  [37] SEQUENCE SIZE (1..MAX) OF DefaultAttribute OPTIONAL
}

DefaultAttribute ::= CHOICE {
  character-set                     [38] INTEGER,
  background-colour                 [39] Colour,
  text-content-hook                 [40] INTEGER,
  text-colour                       [41] Colour,
  font                              [42] FontBody,
  font-attributes                   [43] OCTET STRING,
  interchanged-program-content-hook [44] INTEGER,
  stream-content-hook               [45] INTEGER,
  bitmap-content-hook               [46] INTEGER,
  line-art-content-hook             [47] INTEGER,
  button-ref-colour                 [48] Colour,
  highlight-ref-colour              [49] Colour,
  slider-ref-colour                 [50] Colour
}

FontBody ::= CHOICE {
  direct-font   OCTET STRING,
  indirect-font ObjectReference
}
```

-- <A.4> Scene Class

```
SceneClass ::= SET {
  COMPONENTS OF GroupClass,
  input-event-register    [51] INTEGER,
  scene-coordinate-system [52] SceneCoordinateSystem,
  aspect-ratio            [53] AspectRatio DEFAULT {width 4, height 3},
  moving-cursor           [54] BOOLEAN DEFAULT FALSE,
  next-scenes             [55] SEQUENCE SIZE (1..MAX) OF NextScene OPTIONAL
}

SceneCoordinateSystem ::= SEQUENCE {
  x-scene INTEGER,
  y-scene INTEGER
}

AspectRatio ::= SEQUENCE {
  width  INTEGER,
  height INTEGER
}

NextScene ::= SEQUENCE {
  scene-ref    OCTET STRING,
```

```
  scene-weight INTEGER (0..255)
}
```

-- <A.5> Ingredient Class

```
IngredientClass ::= SET {
  RootClass (WITH COMPONENTS
    {..., external-reference (WITH COMPONENTS {..., object-number (1..MAX)})}),
  initially-active [56] BOOLEAN DEFAULT TRUE,
  content-hook     [57] INTEGER OPTIONAL,
  original-content [58] ContentBody OPTIONAL,
  shared           [59] BOOLEAN DEFAULT FALSE
}

ContentBody ::= CHOICE {
  included-content   OCTET STRING,
  referenced-content ReferencedContent
}

ReferencedContent ::= SEQUENCE {
  content-reference      ContentReference,
  content-size           [60] INTEGER OPTIONAL,
  content-cache-priority [61] INTEGER (0..255) DEFAULT 127
}
```

-- <A.6> Link Class

```
LinkClass ::= SET {
  COMPONENTS OF IngredientClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT}),
  link-condition [62] LinkCondition,
  link-effect    [63] ActionClass
}

LinkCondition ::= SEQUENCE {
  event-source ObjectReference,
  event-type   EventType,
  event-data   EventData OPTIONAL
}

EventType ::= ENUMERATED {
  is-available          (1),
  content-available     (2),
  is-deleted            (3),
  is-running            (4),
  is-stopped            (5),
  user-input            (6),
  anchor-fired          (7),
  timer-fired           (8),
  asynch-stopped        (9),
  interaction-completed (10),
  token-moved-from      (11),
  token-moved-to        (12),
  stream-event          (13),
  stream-playing        (14),
  stream-stopped        (15),
  counter-trigger       (16),
  highlight-on          (17),
  highlight-off         (18),
  cursor-enter          (19),
  cursor-leave          (20),
  is-selected           (21),
  is-deselected         (22),
  test-event            (23),
  first-item-presented  (24),
  last-item-presented   (25),
  head-items            (26),
  tail-items            (27),
  item-selected         (28),
  item-deselected       (29),
  entry-field-full      (30),
  engine-event          (31)
}

EventData ::= CHOICE {
  octetstring OCTET STRING,
  boolean     BOOLEAN,
  integer     INTEGER
}
```

```
-- <A.7> Program Class

ProgramClass ::= SET {
  COMPONENTS OF IngredientClass (WITH COMPONENTS {..., initially-active (FALSE) PRESENT}),
  name                [64] OCTET STRING,
  initially-available [65] BOOLEAN DEFAULT TRUE
}

-- <A.8> ResidentProgram Class

ResidentProgramClass ::= ProgramClass
  (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT})

-- <A.9> RemoteProgram Class

RemoteProgramClass ::= SET {
  COMPONENTS OF ProgramClass (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  program-connection-tag [66] INTEGER OPTIONAL
}

-- <A.10> InterchangedProgram Class

InterchangedProgramClass ::= ProgramClass (WITH COMPONENTS
  {..., original-content PRESENT, content-hook (0)})

-- <A.11> Palette Class

PaletteClass ::= IngredientClass (WITH COMPONENTS
  {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- <A.12> Font Class

FontClass ::= IngredientClass (WITH COMPONENTS
  {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- <A.13> CursorShape Class

CursorShapeClass ::= IngredientClass (WITH COMPONENTS
  {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- <A.14> Variable Class

VariableClass ::= SET {
  COMPONENTS OF IngredientClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT, initially-active (TRUE)}),
  original-value [67] OriginalValue
}

OriginalValue ::= CHOICE {
  boolean           BOOLEAN,
  integer           INTEGER,
  octetstring       OCTET STRING,
  object-reference  [68] ObjectReference,
  content-reference [69] ContentReference
}

-- <A.15> BooleanVariable Class

BooleanVariableClass ::= VariableClass (WITH COMPONENTS
  {..., original-value (WITH COMPONENTS {boolean PRESENT})})

-- <A.16> IntegerVariable Class

IntegerVariableClass ::= VariableClass (WITH COMPONENTS
  {..., original-value (WITH COMPONENTS {integer PRESENT})})

-- <A.17> OctetStringVariable Class

OctetStringVariableClass ::= VariableClass (WITH COMPONENTS
  {..., original-value (WITH COMPONENTS {octetstring PRESENT})})
```

```
-- <A.18> ObjectReferenceVariable Class

ObjectRefVariableClass ::= VariableClass (WITH COMPONENTS
  {..., original-value (WITH COMPONENTS {object-reference PRESENT})})

-- <A.19> ContentReferenceVariable Class

ContentRefVariableClass ::= VariableClass (WITH COMPONENTS
  {..., original-value (WITH COMPONENTS {content-reference PRESENT})})

-- <A.20> Presentable Class

PresentableClass ::= IngredientClass

-- <A.21> TokenManager Class

TokenManagerClass ::= SET {
  movement-table [70] SEQUENCE SIZE (1..MAX) OF Movement OPTIONAL
}

Movement ::= SEQUENCE SIZE (1..MAX) OF INTEGER

-- <A.22> TokenGroup Class

TokenGroupClass ::= SET {
  COMPONENTS OF PresentableClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF TokenManagerClass,
  token-group-items     [71] SEQUENCE SIZE (1..MAX) OF TokenGroupItem,
  no-token-action-slots [72] SEQUENCE SIZE (1..MAX) OF ActionSlot OPTIONAL
}

TokenGroupItem ::= SEQUENCE {
  a-visible    ObjectReference,
  action-slots SEQUENCE SIZE (1..MAX) OF ActionSlot OPTIONAL
}

ActionSlot ::= CHOICE {
  action-class ActionClass,
  null         NULL
}

-- <A.23> ListGroup Class

ListGroupClass ::= SET {
  COMPONENTS OF TokenGroupClass,
  positions          [73] SEQUENCE SIZE (1..MAX) OF XYPosition,
  wrap-around        [74] BOOLEAN DEFAULT FALSE,
  multiple-selection [75] BOOLEAN DEFAULT FALSE
}

-- <A.24> Visible Class

VisibleClass ::= SET {
  COMPONENTS OF PresentableClass,
  original-box-size    [76] OriginalBoxSize,
  original-position    [77] XYPosition DEFAULT {x-position 0, y-position 0},
  original-palette-ref [78] ObjectReference OPTIONAL
}

OriginalBoxSize ::= SEQUENCE {
  x-length INTEGER (0..MAX),
  y-length INTEGER (0..MAX)
}

-- <A.25> Bitmap Class

BitmapClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS {..., original-content PRESENT}),
  tiling               [79] BOOLEAN DEFAULT FALSE,
  original-transparency [80] INTEGER (0..100) DEFAULT 0
}
```

-- <A.26> LineArt Class

```
LineArtClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS {..., original-content PRESENT}),
  bordered-bounding-box    [81] BOOLEAN DEFAULT TRUE,
  original-line-width      [82] INTEGER DEFAULT 1,
  original-line-style      [83] INTEGER {solid(1), dashed(2), dotted(3)} DEFAULT solid,
  original-ref-line-colour [84] Colour OPTIONAL,
  original-ref-fill-colour [85] Colour OPTIONAL
}
```

-- <A.27> Rectangle Class

```
RectangleClass ::= LineArtClass (WITH COMPONENTS
  {..., content-hook ABSENT, original-content ABSENT, bordered-bounding-box ABSENT})
```

-- <A.28> DynamicLineArt Class

```
DynamicLineArtClass ::= LineArtClass (WITH COMPONENTS
  {..., content-hook ABSENT, original-content ABSENT})
```

-- <A.29> Text Class

```
TextClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS {..., original-content PRESENT}),
  original-font            [86] FontBody OPTIONAL,
  font-attributes          [43] OCTET STRING OPTIONAL,
  text-colour              [41] Colour OPTIONAL,
  background-colour        [39] Colour OPTIONAL,
  character-set            [38] INTEGER OPTIONAL,
  horizontal-justification [87] Justification DEFAULT start,
  vertical-justification   [88] Justification DEFAULT start,
  line-orientation         [89] LineOrientation DEFAULT horizontal,
  start-corner             [90] StartCorner DEFAULT upper-left,
  text-wrapping            [91] BOOLEAN DEFAULT FALSE
}

Justification ::= ENUMERATED {start (1), end (2), centre (3), justified (4)}

LineOrientation ::= ENUMERATED {vertical (1), horizontal (2)}

StartCorner ::= ENUMERATED {upper-left (1), upper-right (2), lower-left (3), lower-right (4)}
```

-- <A.30> Stream Class

```
StreamClass ::= SET {
  COMPONENTS OF PresentableClass (WITH COMPONENTS {..., original-content PRESENT}),
  multiplex [92] SEQUENCE SIZE (1..MAX) OF StreamComponent,
  storage   [93] Storage DEFAULT stream,
  looping   [94] INTEGER {infinity(0)} DEFAULT 1
}

StreamComponent ::= CHOICE {
  audio      [95] AudioClass,
  video      [96] VideoClass,
  rtgraphics [97] RTGraphicsClass
}

Storage ::= ENUMERATED {memory (1), stream (2)}
```

-- <A.31> Audio Class

```
AudioClass ::= SET {
  COMPONENTS OF PresentableClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT, shared ABSENT}),
  component-tag   [98] INTEGER,
  original-volume [99] INTEGER DEFAULT 0
}
```

-- <A.32> Video Class

```
VideoClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT, shared ABSENT,
```

```
          original-palette-ref ABSENT}),
  component-tag [98] INTEGER,
  termination   [100] Termination DEFAULT disappear
}

Termination ::= ENUMERATED {freeze (1), disappear (2)}
```

-- <A.33> RTGraphics Class

```
RTGraphicsClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT, shared ABSENT}),
  component-tag [98] INTEGER,
  termination   [100] Termination DEFAULT disappear
}
```

-- <A.34> Interactible Class

```
InteractibleClass ::= SET {
  engine-resp          [101] BOOLEAN DEFAULT TRUE,
  highlight-ref-colour [49] Colour OPTIONAL
}
```

-- <A.35> Slider Class

```
SliderClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF InteractibleClass,
  orientation       [102] Orientation,
  max-value         [103] INTEGER,
  min-value         [104] INTEGER DEFAULT 1,
  initial-value     [105] INTEGER OPTIONAL,
  initial-portion   [106] INTEGER OPTIONAL,
  step-size         [107] INTEGER DEFAULT 1,
  slider-style      [108] SliderStyle DEFAULT normal,
  slider-ref-colour [50] Colour OPTIONAL
}

Orientation ::= ENUMERATED {left (1), right (2), up (3), down (4)}

SliderStyle ::= ENUMERATED {normal (1), thermometer (2), proportional (3)}
```

-- <A.36> EntryField Class

```
EntryFieldClass ::= SET {
  COMPONENTS OF TextClass,
  COMPONENTS OF InteractibleClass,
  input-type     [109] InputType DEFAULT any,
  char-list      [110] OCTET STRING OPTIONAL,
  obscured-input [111] BOOLEAN DEFAULT FALSE,
  max-length     [112] INTEGER DEFAULT 0
}

InputType ::= ENUMERATED {alpha (1), numeric (2), any (3), listed (4)}
```

-- <A.37> HyperText Class

```
HyperTextClass ::= SET {
  COMPONENTS OF TextClass,
  COMPONENTS OF InteractibleClass
}
```

-- <A.38> Button Class

```
ButtonClass ::= SET {
  COMPONENTS OF VisibleClass (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF InteractibleClass,
  button-ref-colour [48] Colour OPTIONAL
}
```

-- <A.39> Hotspot Class

```
HotspotClass ::= ButtonClass
```

-- <A.40> PushButton Class

```
PushButtonClass ::= SET {
  COMPONENTS OF ButtonClass,
  original-label [113] OCTET STRING OPTIONAL,
  character-set  [38] INTEGER OPTIONAL
}
```

-- <A.41> SwitchButton Class

```
SwitchButtonClass ::= SET {
  COMPONENTS OF PushButtonClass,
  button-style [114] ButtonStyle
}

ButtonStyle ::= ENUMERATED {pushbutton (1), radiobutton (2), checkbox (3)}
```

-- <A.42> Applet Class

```
AppletClass ::= SET {
  COMPONENTS OF InterchangedProgramClass,
  COMPONENTS OF InteractibleClass,
  original-box-size [76] OriginalBoxSize,
  original-position [77] XYPosition DEFAULT {x-position 0, y-position 0}
}
```

-- <A.43> Action Class

```
ActionClass ::= SEQUENCE SIZE (1..MAX) OF ElementaryAction

ElementaryAction ::= CHOICE {
  activate                [115] GenericObjectReference,
  add                     [116] Add,
  add-item                [117] AddItem,
  append                  [118] Append,
  bring-to-front          [119] GenericObjectReference,
  call                    [120] Call,
  call-action-slot        [121] CallActionSlot,
  clear                   [122] GenericObjectReference,
  clone                   [123] Clone,
  close-connection        [124] CloseConnection,
  deactivate              [125] GenericObjectReference,
  del-item                [126] DelItem,
  deselect                [127] GenericObjectReference,
  deselect-item           [128] DeselectItem,
  divide                  [129] Divide,
  draw-arc                [130] DrawArc,
  draw-line               [131] DrawLine,
  draw-oval               [132] DrawOval,
  draw-polygon            [133] DrawPolygon,
  draw-polyline           [134] DrawPolyline,
  draw-rectangle          [135] DrawRectangle,
  draw-sector             [136] DrawSector,
  fork                    [137] Fork,
  get-availability-status [138] GetAvailabilityStatus,
  get-box-size            [139] GetBoxSize,
  get-cell-item           [140] GetCellItem,
  get-cursor-position     [141] GetCursorPosition,
  get-engine-support      [142] GetEngineSupport,
  get-entry-point         [143] GetEntryPoint,
  get-fill-colour         [144] GetFillColour,
  get-first-item          [145] GetFirstItem,
  get-highlight-status    [146] GetHighlightStatus,
  get-interaction-status  [147] GetInteractionStatus,
  get-item-status         [148] GetItemStatus,
  get-label               [149] GetLabel,
  get-last-anchor-fired   [150] GetLastAnchorFired,
  get-line-colour         [151] GetLineColour,
  get-line-style          [152] GetLineStyle,
  get-line-width          [153] GetLineWidth,
  get-list-item           [154] GetListItem,
  get-list-size           [155] GetListSize,
  get-overwrite-mode      [156] GetOverwriteMode,
  get-portion             [157] GetPortion,
  get-position            [158] GetPosition,
  get-running-status      [159] GetRunningStatus,
  get-selection-status    [160] GetSelectionStatus,
```

```
  get-slider-value          [161] GetSliderValue,
  get-text-content          [162] GetTextContent,
  get-text-data             [163] GetTextData,
  get-token-position        [164] GetTokenPosition,
  get-volume                [165] GetVolume,
  invoke                    [238] Invoke,
  launch                    [166] GenericObjectReference,
  lock-screen               [167] GenericObjectReference,
  modulo                    [168] Modulo,
  move                      [169] Move,
  move-to                   [170] MoveTo,
  multiply                  [171] Multiply,
  open-connection           [172] OpenConnection,
  preload                   [173] GenericObjectReference,
  put-before                [174] PutBefore,
  put-behind                [175] PutBehind,
  quit                      [176] GenericObjectReference,
  read-persistent           [177] ReadPersistent,
  run                       [178] GenericObjectReference,
  scale-bitmap              [179] ScaleBitmap,
  scale-video               [180] ScaleVideo,
  scroll-items              [181] ScrollItems,
  select                    [182] GenericObjectReference,
  select-item               [183] SelectItem,
  send-event                [184] SendEvent,
  send-to-back              [185] GenericObjectReference,
  set-box-size              [186] SetBoxSize,
  set-cache-priority        [187] SetCachePriority,
  set-counter-end-position  [188] SetCounterEndPosition,
  set-counter-position      [189] SetCounterPosition,
  set-counter-trigger       [190] SetCounterTrigger,
  set-cursor-position       [191] SetCursorPosition,
  set-cursor-shape          [192] SetCursorShape,
  set-data                  [193] SetData,
  set-entry-point           [194] SetEntryPoint,
  set-fill-colour           [195] SetFillColour,
  set-first-item            [196] SetFirstItem,
  set-font-ref              [197] SetFontRef,
  set-highlight-status      [198] SetHighlightStatus,
  set-interaction-status    [199] SetInteractionStatus,
  set-label                 [200] SetLabel,
  set-line-colour           [201] SetLineColour,
  set-line-style            [202] SetLineStyle,
  set-line-width            [203] SetLineWidth,
  set-overwrite-mode        [204] SetOverwriteMode,
  set-palette-ref           [205] SetPaletteRef,
  set-portion               [206] SetPortion,
  set-position              [207] SetPosition,
  set-slider-value          [208] SetSliderValue,
  set-speed                 [209] SetSpeed,
  set-timer                 [210] SetTimer,
  set-transparency          [211] SetTransparency,
  set-variable              [212] SetVariable,
  set-volume                [213] SetVolume,
  spawn                     [214] GenericObjectReference,
  step                      [215] Step,
  stop                      [216] GenericObjectReference,
  store-persistent          [217] StorePersistent,
  subtract                  [218] Subtract,
  test-variable             [219] TestVariable,
  toggle                    [220] GenericObjectReference,
  toggle-item               [221] ToggleItem,
  transition-to             [222] TransitionTo,
  unload                    [223] GenericObjectReference,
  unlock-screen             [224] GenericObjectReference
}

Add ::= SEQUENCE {
  target GenericObjectReference,
  value  GenericInteger
}

AddItem ::= SEQUENCE {
  target           GenericObjectReference,
  item-index       GenericInteger,
  visible-reference GenericObjectReference
}
```

```
Append ::= SEQUENCE {
  target       GenericObjectReference,
  append-value GenericOctetString
}

Call ::= SEQUENCE {
  target         GenericObjectReference,
  call-succeeded ObjectReference,
  parameters     SEQUENCE SIZE (1..MAX) OF Parameter OPTIONAL
}

CallActionSlot ::= SEQUENCE {
  target GenericObjectReference,
  index  GenericInteger
}

Clone ::= SEQUENCE {
  target        GenericObjectReference,
  clone-ref-var ObjectReference
}

CloseConnection ::= SEQUENCE {
  target         GenericObjectReference,
  connection-tag GenericInteger
}

DelItem ::= SEQUENCE {
  target            GenericObjectReference,
  visible-reference GenericObjectReference
}

DeselectItem ::= SEQUENCE {
  target     GenericObjectReference,
  item-index GenericInteger
}

Divide ::= SEQUENCE {
  target GenericObjectReference,
  value  GenericInteger
}

DrawArc ::= SEQUENCE {
  target         GenericObjectReference,
  x              GenericInteger,
  y              GenericInteger,
  ellipse-width  GenericInteger,
  ellipse-height GenericInteger,
  start-angle    GenericInteger,
  arc-angle      GenericInteger
}

DrawLine ::= SEQUENCE {
  target GenericObjectReference,
  x1     GenericInteger,
  y1     GenericInteger,
  x2     GenericInteger,
  y2     GenericInteger
}

DrawOval ::= SEQUENCE {
  target         GenericObjectReference,
  x              GenericInteger,
  y              GenericInteger,
  ellipse-width  GenericInteger,
  ellipse-height GenericInteger
}

DrawPolygon ::= SEQUENCE {
  target    GenericObjectReference,
  pointlist SEQUENCE SIZE (1..MAX) OF Point
}

DrawPolyline ::= SEQUENCE {
  target    GenericObjectReference,
  pointlist SEQUENCE SIZE (1..MAX) OF Point
}

DrawRectangle ::= SEQUENCE {
  target GenericObjectReference,
```

```
  x1     GenericInteger,
  y1     GenericInteger,
  x2     GenericInteger,
  y2     GenericInteger
}

DrawSector ::= SEQUENCE {
  target         GenericObjectReference,
  x              GenericInteger,
  y              GenericInteger,
  ellipse-width  GenericInteger,
  ellipse-height GenericInteger,
  start-angle    GenericInteger,
  arc-angle      GenericInteger
}

Fork ::= SEQUENCE {
  target          GenericObjectReference,
  fork-succeeded ObjectReference,
  parameters      SEQUENCE SIZE (1..MAX) OF Parameter OPTIONAL
}

GetAvailabilityStatus ::= SEQUENCE {
  target                  GenericObjectReference,
  availability-status-var ObjectReference
}

GetBoxSize ::= SEQUENCE {
  target         GenericObjectReference,
  x-box-size-var ObjectReference,
  y-box-size-var ObjectReference
}

GetCellItem ::= SEQUENCE {
  target       GenericObjectReference,
  cell-index   GenericInteger,
  item-ref-var ObjectReference
}

GetCursorPosition ::= SEQUENCE {
  target GenericObjectReference,
  x-out  ObjectReference,
  y-out  ObjectReference
}

GetEngineSupport ::= SEQUENCE {
  target  GenericObjectReference,
  feature GenericOctetString,
  answer  ObjectReference
}

GetEntryPoint ::= SEQUENCE {
  target          GenericObjectReference,
  entry-point-var ObjectReference
}

GetFillColour ::= SEQUENCE {
  target          GenericObjectReference,
  fill-colour-var ObjectReference
}

GetFirstItem ::= SEQUENCE {
  target          GenericObjectReference,
  first-item-var  ObjectReference
}

GetHighlightStatus ::= SEQUENCE {
  target               GenericObjectReference,
  highlight-status-var ObjectReference
}

GetInteractionStatus ::= SEQUENCE {
  target                 GenericObjectReference,
  interaction-status-var ObjectReference
}

GetItemStatus ::= SEQUENCE {
  target          GenericObjectReference,
```

```
  item-index      GenericInteger,
  item-status-var ObjectReference
}

GetLabel ::= SEQUENCE {
  target    GenericObjectReference,
  label-var ObjectReference
}

GetLastAnchorFired ::= SEQUENCE {
  target                GenericObjectReference,
  last-anchor-fired-var ObjectReference
}

GetLineColour ::= SEQUENCE {
  target          GenericObjectReference,
  line-colour-var ObjectReference
}

GetLineStyle ::= SEQUENCE {
  target         GenericObjectReference,
  line-style-var ObjectReference
}

GetLineWidth ::= SEQUENCE {
  target         GenericObjectReference,
  line-width-var ObjectReference
}

GetListItem ::= SEQUENCE {
  target       GenericObjectReference,
  item-index   GenericInteger,
  item-ref-var ObjectReference
}

GetListSize ::= SEQUENCE {
  target   GenericObjectReference,
  size-var ObjectReference
}

GetOverwriteMode ::= SEQUENCE {
  target             GenericObjectReference,
  overwrite-mode-var ObjectReference
}

GetPortion ::= SEQUENCE {
  target      GenericObjectReference,
  portion-var ObjectReference
}

GetPosition ::= SEQUENCE {
  target         GenericObjectReference,
  x-position-var ObjectReference,
  y-position-var ObjectReference
}

GetRunningStatus ::= SEQUENCE {
  target             GenericObjectReference,
  running-status-var ObjectReference
}

GetSelectionStatus ::= SEQUENCE {
  target               GenericObjectReference,
  selection-status-var ObjectReference
}

GetSliderValue ::= SEQUENCE {
  target           GenericObjectReference,
  slider-value-var ObjectReference
}

GetTextContent ::= SEQUENCE {
  target           GenericObjectReference,
  text-content-var ObjectReference
}

GetTextData ::= SEQUENCE {
  target        GenericObjectReference,
```

```
  text-data-var ObjectReference
}

GetTokenPosition ::= SEQUENCE {
  target             GenericObjectReference,
  token-position-var ObjectReference
}

GetVolume ::= SEQUENCE {
  target     GenericObjectReference,
  volume-var ObjectReference
}

Invoke ::= SEQUENCE {
  target           GenericObjectReference,
  invoke-succeeded ObjectReference,
  method           GenericOctetString,
  parameters       SEQUENCE SIZE (1..MAX) OF Parameter OPTIONAL
}

Modulo ::= SEQUENCE {
  target GenericObjectReference,
  value  GenericInteger
}

Move ::= SEQUENCE {
  target              GenericObjectReference,
  movement-identifier GenericInteger
}

MoveTo ::= SEQUENCE {
  target GenericObjectReference,
  index  GenericInteger
}

Multiply ::= SEQUENCE {
  target GenericObjectReference,
  value  GenericInteger
}

OpenConnection ::= SEQUENCE {
  target         GenericObjectReference,
  open-succeeded ObjectReference,
  protocol       GenericOctetString,
  address        GenericOctetString,
  connection-tag GenericInteger
}

PutBefore ::= SEQUENCE {
  target            GenericObjectReference,
  reference-visible GenericObjectReference
}

PutBehind ::= SEQUENCE {
  target            GenericObjectReference,
  reference-visible GenericObjectReference
}

ReadPersistent ::= SEQUENCE {
  target         GenericObjectReference,
  read-succeeded ObjectReference,
  out-variables  SEQUENCE SIZE (1..MAX) OF ObjectReference,
  in-file-name   GenericOctetString
}

ScaleBitmap ::= SEQUENCE {
  target  GenericObjectReference,
  x-scale GenericInteger,
  y-scale GenericInteger
}

ScaleVideo ::= SEQUENCE {
  target  GenericObjectReference,
  x-scale GenericInteger,
  y-scale GenericInteger
}

ScrollItems ::= SEQUENCE {
```

```
  target          GenericObjectReference,
  items-to-scroll GenericInteger
}

SelectItem ::= SEQUENCE {
  target     GenericObjectReference,
  item-index GenericInteger
}

SendEvent ::= SEQUENCE {
  target               GenericObjectReference,
  emulated-event-source GenericObjectReference,
  emulated-event-type   EventType,
  emulated-event-data   EmulatedEventData OPTIONAL
}

SetBoxSize ::= SEQUENCE {
  target         GenericObjectReference,
  x-new-box-size GenericInteger,
  y-new-box-size GenericInteger
}

SetCachePriority ::= SEQUENCE {
  target             GenericObjectReference,
  new-cache-priority GenericInteger
}

SetCounterEndPosition ::= SEQUENCE {
  target                   GenericObjectReference,
  new-counter-end-position GenericInteger
}

SetCounterPosition ::= SEQUENCE {
  target               GenericObjectReference,
  new-counter-position GenericInteger
}

SetCounterTrigger ::= SEQUENCE {
  target             GenericObjectReference,
  trigger-identifier GenericInteger,
  new-counter-value  GenericInteger OPTIONAL
}

SetCursorPosition ::= SEQUENCE {
  target   GenericObjectReference,
  x-cursor GenericInteger,
  y-cursor GenericInteger
}

SetCursorShape ::= SEQUENCE {
  target           GenericObjectReference,
  new-cursor-shape GenericObjectReference OPTIONAL
}

SetData ::= SEQUENCE {
  target      GenericObjectReference,
  new-content NewContent
}

SetEntryPoint ::= SEQUENCE {
  target          GenericObjectReference,
  new-entry-point GenericInteger
}

SetFillColour ::= SEQUENCE {
  target          GenericObjectReference,
  new-fill-colour NewColour OPTIONAL
}

SetFirstItem ::= SEQUENCE {
  target         GenericObjectReference,
  new-first-item GenericInteger
}

SetFontRef ::= SEQUENCE {
  target   GenericObjectReference,
  new-font NewFont
}
```

```
SetHighlightStatus ::= SEQUENCE {
  target               GenericObjectReference,
  new-highlight-status GenericBoolean
}

SetInteractionStatus ::= SEQUENCE {
  target                 GenericObjectReference,
  new-interaction-status GenericBoolean
}

SetLabel ::= SEQUENCE {
  target    GenericObjectReference,
  new-label GenericOctetString
}

SetLineColour ::= SEQUENCE {
  target          GenericObjectReference,
  new-line-colour NewColour
}

SetLineStyle ::= SEQUENCE {
  target         GenericObjectReference,
  new-line-style GenericInteger
}

SetLineWidth ::= SEQUENCE {
  target         GenericObjectReference,
  new-line-width GenericInteger
}

SetOverwriteMode ::= SEQUENCE {
  target             GenericObjectReference,
  new-overwrite-mode GenericBoolean
}

SetPaletteRef ::= SEQUENCE {
  target          GenericObjectReference,
  new-palette-ref GenericObjectReference
}

SetPortion ::= SEQUENCE {
  target      GenericObjectReference,
  new-portion GenericInteger
}

SetPosition ::= SEQUENCE {
  target         GenericObjectReference,
  new-x-position GenericInteger,
  new-y-position GenericInteger
}

SetSliderValue ::= SEQUENCE {
  target          GenericObjectReference,
  new-slider-value GenericInteger
}

SetSpeed ::= SEQUENCE {
  target    GenericObjectReference,
  new-speed Rational
}

SetTimer ::= SEQUENCE {
  target    GenericObjectReference,
  timer-id  GenericInteger,
  new-timer NewTimer OPTIONAL
}

NewTimer ::= SEQUENCE {
  timer-value   GenericInteger,
  absolute-time GenericBoolean OPTIONAL
}

SetTransparency ::= SEQUENCE {
  target           GenericObjectReference,
  new-transparency GenericInteger
}
```

```
SetVariable ::= SEQUENCE {
  target             GenericObjectReference,
  new-variable-value NewVariableValue
}

SetVolume ::= SEQUENCE {
  target     GenericObjectReference,
  new-volume GenericInteger
}

Step ::= SEQUENCE {
  target      GenericObjectReference,
  nb-of-steps GenericInteger
}

StorePersistent ::= SEQUENCE {
  target          GenericObjectReference,
  store-succeeded ObjectReference,
  in-variables    SEQUENCE SIZE (1..MAX) OF ObjectReference,
  out-file-name   GenericOctetString
}

Subtract ::= SEQUENCE {
  target GenericObjectReference,
  value  GenericInteger
}

TestVariable ::= SEQUENCE {
  target           GenericObjectReference,
  operator         GenericInteger,
  comparison-value ComparisonValue
}

ToggleItem ::= SEQUENCE {
  target     GenericObjectReference,
  item-index GenericInteger
}

TransitionTo ::= SEQUENCE {
  target                GenericObjectReference,
  connection-tag-or-null ConnectionTagOrNull,
  transition-effect      GenericInteger OPTIONAL
}

ConnectionTagOrNull ::= CHOICE {
  connection-tag GenericInteger,
  null           NULL
}

ComparisonValue ::= CHOICE {
  new-generic-boolean          [225] GenericBoolean,
  new-generic-integer          [226] GenericInteger,
  new-generic-octetstring      [227] GenericOctetString,
  new-generic-object-reference [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

EmulatedEventData ::= CHOICE {
  new-generic-boolean      [225] GenericBoolean,
  new-generic-integer      [226] GenericInteger,
  new-generic-octet-string [227] GenericOctetString
}

NewColour ::= CHOICE {
  new-colour-index    [230] GenericInteger,
  new-absolute-colour [231] GenericOctetString
}

NewContent ::= CHOICE {
  new-included-content   GenericOctetString,
  new-referenced-content NewReferencedContent
}

NewFont ::= CHOICE {
  new-font-name      [232] GenericOctetString,
  new-font-reference [233] GenericObjectReference
}
```

```
NewReferencedContent ::= SEQUENCE {
  generic-content-reference  GenericContentReference,
  new-content-size           [234] NewContentSize,
  new-content-cache-priority [235] GenericInteger OPTIONAL
}

NewContentSize ::= CHOICE {
  content-size GenericInteger,
  null         NULL
}

NewVariableValue ::= CHOICE {
  new-generic-boolean           [225] GenericBoolean,
  new-generic-integer           [226] GenericInteger,
  new-generic-octetstring       [227] GenericOctetString,
  new-generic-object-reference  [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

Parameter ::= CHOICE {
  new-generic-boolean           [225] GenericBoolean,
  new-generic-integer           [226] GenericInteger,
  new-generic-octetstring       [227] GenericOctetString,
  new-generic-object-reference  [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

Point ::= SEQUENCE {
  x GenericInteger,
  y GenericInteger
}

Rational ::= SEQUENCE {
  numerator   GenericInteger,
  denominator GenericInteger OPTIONAL
}
```

-- <A.44> Miscellaneous data types

```
ObjectReference ::= CHOICE {
  external-reference ExternalReference,
  internal-reference INTEGER (1..MAX)
}

ExternalReference ::= SEQUENCE {
  group-identifier OCTET STRING,
  object-number    INTEGER (0..MAX)
}

IndirectReference ::= [236] ObjectReference

ContentReference ::= OCTET STRING

GenericObjectReference ::= CHOICE {
  direct-reference   ObjectReference,
  indirect-reference IndirectReference
}

GenericContentReference ::= CHOICE {
  content-reference  [69] ContentReference,
  indirect-reference IndirectReference
}

GenericInteger ::= CHOICE {
  integer            INTEGER,
  indirect-reference IndirectReference
}

GenericBoolean ::= CHOICE {
  boolean            BOOLEAN,
  indirect-reference IndirectReference
}

GenericOctetString ::= CHOICE {
  octetstring        OCTET STRING,
  indirect-reference IndirectReference
}
```

```
Colour ::= CHOICE {
  colour-index    INTEGER,
  absolute-colour OCTET STRING
}

XYPosition ::= SEQUENCE {
  x-position INTEGER,
  y-position INTEGER
}

END
```

Annex B
(normative)

Textual notation

This Annex describes the textual notation for the syntax of MHEG-6 objects. This is an alternative representation to the ASN.1 notation defined in Annex A.

The textual notation has been introduced especially for the purposes of designing and testing objects. It is equivalent to the ASN.1 notation, and both notations can be easily derived from each other.

The syntax below is compatible with that defined by Annex B of ISO/IEC 13522-5. So any MHEG-5 object encoded according to Annex B of ISO/IEC 13522-5 is also an MHEG-6 object that conforms to the syntax in this Annex.

B.1 General Definitions

B.1.1 Code

The textual notation shall use the subset of the ISO/IEC 646:1991 code set defined as the range of characters from 0x20 (SP) to 0x7e (~), plus 0x09 (HT), 0x0a (LF), 0x0c (FF), and 0x0d (CR).

Other characters shall not be used.

NOTE 1 Despite the fact that the textual notation limits the character codes to be used, contents of 8-bit data can be encoded by means of QPRINTABLE (see B.3.4), BASE64 (see B.3.5) and external contents referenced by ContentReference.

NOTE 2 An application domain may extend the character codes to be used in the textual notation as far as it does not violate the grammar. For example, characters from 0x80 to 0xfe might also be allowed for STRING and comments.

B.1.2 Delimiter

0x09 (HT), 0x0a (LF), 0x0c (FF), 0x0d (CR) and 0x20 (SP) are called delimiters.

The grammar described by the textual notation is word-based. A word is either a parenthesis ("(" or ")"), a brace ("{" or "}"), a tag (see B.1.4) or a terminal symbol (see B.3). Any number of delimiters may be inserted between any two adjacent words, without changing the interpretation of words. However, at least one delimiter shall exist between any two terminal symbols and between any tag and any terminal symbol, since they would otherwise be interpreted as a single terminal or a single tag.

B.1.3 Comment

// (0x2f 0x2f) which is not within a STRING, QPRINTABLE and BASE64 (see B.3.3, B.3.4 and B.3.5) is used to indicate the start of a comment. All characters between such // (including the //) and the next occurrence of a 0x0a (LF), 0x0c (FF) or 0x0d (CR) shall be ignored.

NOTE 0x09 (HT) and 0x20 (SP) do not indicate the end of comments.

B.1.4 Tag

A token starting with : (0x3a) is called "tag". A tag is preceded by { (0x7b) when it is used at the beginning of MHEG-5 objects. A tag is used to distinguish the MHEG-5 objects and their associated attribute values, in general. Tags are case-insensitive, e.g. ":Root", ":root", ":ROOT", ":rOOt" and so on are all same. However, in this textual notation, some combination of upper case and lower case in tags are used for easier understanding and improved readability.

B.2 Definitions of Symbols

Table B.1 shows the symbols used in the textual notation and their meanings.

Table B.1 - Definitions of Symbols in Textual Notation

Symbol	Definition
::=	Is defined to be.
|	Alternative.
<">	Double quote mark (0x22).
"text"	Literals enclosed in double quotes.
<text>	Plain text description explaining the codes to be here.
*	The preceding syntactic unit may be repeated zero or more times.
+	The preceding syntactic unit may be repeated one or more times.
[]	The enclosed syntactic unit is optional. It may occur zero or one times.
.	End of clause.

B.3 Terminal Symbols

All the terminal symbols used in the textual notation are defined as follows.

B.3.1 INTEGER

A decimal or positive hexadecimal integer value.

Definition:

```
INTEGER     ::= DECINT | HEXINT | "0" .
DECINT      ::= ["-"] DIGIT [DIGIT0]* .
DIGIT       ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
DIGIT0      ::= DIGIT | "0" .
HEXINT      ::= HEXPREFIX HEXDIGIT0 [HEXDIGIT0]* .
HEXPREFIX   ::= "0x" | "0X" .
HEXDIGIT0   ::= DIGIT | "0" | "a" | "b" | "c" | "d" | "e" | "f" | "A" | "B" |
                "C" | "D" | "E" | "F" .
```

B.3.2 BOOLEAN

A Boolean value may be either true or false. The BOOLEAN terminal is case-insensitive, i.e. "True", "TRUE" and "true" are equivalent, "False", "FALSE" and "false" are equivalent.

Definition:

```
BOOLEAN     ::= "true" | "false" .
```

B.3.3 STRING

A string value enclosed in the double quotes may contain an arbitrary number of printable characters (from 0x20 to 0x7e). A double quote (0x22) within a STRING shall be encoded as \" (0x5c 0x22). A backslash (0x5c) shall be encoded as \\ (0x5c 0x5c).

No line breaks should be included in STRING: multi-line text content shall be encoded using QPRINTABLE or BASE64.

Definition:

```
STRING      ::= <"> STRINGCHAR* <"> .
STRINGCHAR  ::= <any single printable character except <"> and "\"> | "\"" |
                "\\" .
```

B.3.4 QPRINTABLE

A string value enclosed in the single quotes shall contain a Quoted-Printable encoded content as defined in RFC-1521. However, ' (0x27) shall be encoded as =27. The lower case characters such as a, b, c, d, e and f may be used as a general 8-bit representation defined by section 5.1 rule #1 in RFC-1521. The number of characters in a line is not limited. Line breaks need not be converted to CR (0x0d)/LF (0x0a), however, at least one of LF (0x0a), FF (0x0c) and CR (0x0d) shall be used to represent a line break.

Definition:

```
QPRINTABLE        ::= "'" QPRINTABLECHAR* "'" .
QPRINTABLECHAR    ::= <a character as defined above> |
                      <encoding sequence of a character as defined above> .
```

B.3.5 BASE64

A string value enclosed in the back quotes shall contain a BASE64 encoded content as defined in RFC-1521. However, the number of characters in a line is not limited in this standard. The encoded BASE64 content may be split into several segments by at least one of LF (0x0a), FF (0x0c) or CR (0x0d). However, these characters shall be ignored, and the input BASE64 encoded segments shall be handled as if they were in one line.

Definition:

```
BASE64      ::= "`" BASE64CHAR* "`" .
BASE64CHAR  ::= <an encoding sequence of a character as defined above> .
```

B.3.6 Null

Null represents a special terminal whose semantics depends on the MHEG-5 objects definition. The Null terminal is case-insensitive, i.e. "NULL" and "null" are equivalent.

Definition:

```
Null  ::= "NULL" .
```

B.3.7 Enumeration Values

A terminal word starting with an alphabet character is called "enumeration value" (all enumeration values are enclosed within two double quotations in the following grammar). An enumeration value is used as one of the terminal symbols which represents specific semantics depending on its usage. All enumeration values are case-insensitive, i.e. "IsAvailable" and "isavailable" are equivalent.

B.4 MHEG-5 Object Definitions

The textual notation of MHEG-5 objects is defined as follows.

Table B.2 summarises the abbreviations used in tags.

Table B.2 - Tag abbreviations

Bordered Bounding Box	BBBox
Content Cache Priority	CCPriority
Content Hook	CHook
Coordinate System	CS
Generic	G
Group Cache Priority	GCPriority
Horizontal Justification	HJustification
Identifier	ID
Information	Info
Interchanged	Interchg
OctetString	OString
Original	Orig
Program	Prg
Reference	Ref
Register	Reg
Standard	Std
Variable	Var (except for elementary action tags)
Vertical Justification	VJustification


```
--Copyright statement:
--© International Organisation for Standardisation 1998
--Permission to copy in any form is granted for use with conforming MHEG-6 engines and applications
--provided this notice is included in all copies
```

B.4.1 Root Class

```
Root                  ::= ObjectIdentifier .
ObjectIdentifier      ::= ObjectReference .
```

B.4.2 Group Class

```
Group                 ::= Root [StandardIdentifier]
                          [StandardVersion] [ObjectInformation]
                          [OnStartUp] [OnCloseDown]
                          [OriginalGroupCachePriority] [Items] .
StandardIdentifier    ::= ":StdID" JointIsoItuIdentifier
                          MHEGStandardIdentifier .
JointIsoItuIdentifier ::= INTEGER .
MHEGStandardIdentifier::= INTEGER .
StandardVersion       ::= ":StdVersion" INTEGER .
ObjectInformation     ::= ":ObjectInfo" OctetString .
OnStartUp             ::= ":OnStartUp" ActionClass .
OnCloseDown           ::= ":OnCloseDown" ActionClass .
OriginalGroupCachePriority ::= ":OrigGCPriority" INTEGER .
Items                 ::= ":Items" "(" GroupItem+ ")" .
GroupItem             ::= ResidentProgramClass |
                          RemoteProgramClass |
                          InterchangedProgramClass |
                          PaletteClass |
                          FontClass |
                          CursorShapeClass |
                          BooleanVariableClass |
                          IntegerVariableClass |
                          OctetStringVariableClass |
                          ObjectRefVariableClass |
                          ContentRefVariableClass |
                          LinkClass |
```

```
                        StreamClass |
                        BitmapClass |
                        LineArtClass |
                        DynamicLineArtClass |
                        RectangleClass |
                        HotspotClass |
                        SwitchButtonClass |
                        PushButtonClass |
                        TextClass |
                        EntryFieldClass |
                        HyperTextClass |
                        SliderClass |
                        TokenGroupClass |
                        ListGroupClass .
```

B.4.3 Application Class

```
ApplicationClass        ::= "{:Application" Group
                            [OnSpawnCloseDown] [OnRestart]
                            [DefaultAttributes] "}" .
OnSpawnCloseDown        ::= ":OnSpawnCloseDown" ActionClass .
OnRestart               ::= ":OnRestart" ActionClass .
DefaultAttributes       ::= DefaultAttribute+ .
DefaultAttribute        ::= CharacterSet | BackgroundColour
                            | TextContentHook
                            | TextColour | Font
                            | FontAttributes
                            | InterchangedProgramContentHook
                            | StreamContentHook
                            | BitmapContentHook
                            | LineArtContentHook | ButtonRefColour
                            | HighlightRefColour | SliderRefColour .
CharacterSet            ::= ":CharacterSet" INTEGER .
BackgroundColour        ::= ":BackgroundColour" Colour .
TextContentHook         ::= ":TextCHook" INTEGER .
TextColour              ::= ":TextColour" Colour .
Font                    ::= ":Font" FontBody .
FontBody                ::= DirectFont | IndirectFont .
DirectFont              ::= OctetString .
IndirectFont            ::= ObjectReference .
FontAttributes          ::= ":FontAttributes" OctetString .
InterchangedProgramContentHook ::= ":InterchgPrgCHook" INTEGER .
StreamContentHook       ::= ":StreamCHook" INTEGER .
BitmapContentHook       ::= ":BitmapCHook" INTEGER .
LineArtContentHook      ::= ":LineArtCHook" INTEGER .
ButtonRefColour         ::= ":ButtonRefColour" Colour .
HighlightRefColour      ::= ":HighlightRefColour" Colour .
SliderRefColour         ::= ":SliderRefColour" Colour .
```

B.4.4 Scene Class

```
SceneClass              ::= "{:Scene" Group InputEventRegister
                            SceneCoordinateSystem [AspectRatio]
                            [MovingCursor] [NextScenes] "}" .
InputEventRegister      ::= ":InputEventReg" INTEGER .
SceneCoordinateSystem   ::= ":SceneCS" XScene YScene .
XScene                  ::= INTEGER .
YScene                  ::= INTEGER .
AspectRatio             ::= ":AspectRatio" Width Height .
Width                   ::= INTEGER .
Height                  ::= INTEGER .
MovingCursor            ::= ":MovingCursor" BOOLEAN .
NextScenes              ::= ":NextScenes" "(" NextScene+ ")" .
NextScene               ::= "(" SceneRef SceneWeight ")" .
SceneRef                ::= OctetString .
SceneWeight             ::= INTEGER .
```

B.4.5 Ingredient Class

```
Ingredient              ::= Root [InitiallyActive] [ContentHook]
                            [OriginalContent] [Shared] .
InitiallyActive         ::= ":InitiallyActive" BOOLEAN .
ContentHook             ::= ":CHook" INTEGER .
OriginalContent         ::= ":OrigContent" ContentBody .
ContentBody             ::= IncludedContent | ReferencedContent .
IncludedContent         ::= OctetString .
```

```
ReferencedContent      ::= ":ContentRef" "(" ContentReference
                           [ContentSize] [ContentCachePriority]
                           ")" .
ContentSize            ::= ":ContentSize" INTEGER .
ContentCachePriority   ::= ":CCPriority" INTEGER .
Shared                 ::= ":Shared" BOOLEAN .
```

B.4.6 Link Class

```
LinkClass              ::= "{:Link" Ingredient LinkCondition
                           LinkEffect "}" .
LinkCondition          ::= EventSource EventType [EventData] .
EventSource            ::= ":EventSource" ObjectReference .
EventType              ::= ":EventType" EventTypeEnum .
EventTypeEnum          ::= "IsAvailable" | "ContentAvailable"
                           | "IsDeleted" | "IsRunning"
                           | "IsStopped" | "UserInput"
                           | "AnchorFired" | "TimerFired"
                           | "AsynchStopped" | "InteractionCompleted"
                           | "TokenMovedFrom" | "TokenMovedTo"
                           | "StreamEvent" | "StreamPlaying"
                           | "StreamStopped" | "CounterTrigger"
                           | "HighlightOn" | "HighlightOff"
                           | "CursorEnter" | "CursorLeave"
                           | "IsSelected" | "IsDeselected"
                           | "TestEvent" | "FirstItemPresented"
                           | "LastItemPresented" | "HeadItems"
                           | "TailItems" | "ItemSelected"
                           | "ItemDeselected" | "EntryFieldFull" | «EngineEvent» .
EventData              ::= ":EventData" EventDataBody .
EventDataBody          ::= OctetString | BOOLEAN | INTEGER .
LinkEffect             ::= ":LinkEffect" ActionClass .
```

B.4.7 Program Class

```
Program                ::= Ingredient Name [InitiallyAvailable] .
Name                   ::= ":Name" OctetString .
InitiallyAvailable     ::= ":InitiallyAvailable" BOOLEAN .
```

B.4.8 ResidentProgram Class

```
ResidentProgramClass   ::= "{:ResidentPrg" Program "}" .
```

B.4.9 RemoteProgram Class

```
RemoteProgramClass     ::= "{:RemotePrg" Program
                           [ProgramConnectionTag] "}" .
ProgramConnectionTag   ::= ":ConnectionTag" INTEGER .
```

B.4.10 InterchangedProgram Class

```
InterchangedProgramClass      ::= "{:InterchgPrg" Program "}" .
```

B.4.11 Palette Class

```
PaletteClass           ::= "{:Palette" Ingredient "}" .
```

B.4.12 Font Class

```
FontClass              ::= "{:Font" Ingredient "}" .
```

B.4.13 CursorShape Class

```
CursorShapeClass       ::= "{:CursorShape" Ingredient "}" .
```

B.4.14 Variable Class

```
Variable               ::= Ingredient OriginalValue .
OriginalValue          ::= ":OrigValue" OriginalValueBody .
OriginalValueBody      ::= BOOLEAN | INTEGER | OctetString
```

```
                        | ObjectReferenceValue
                        | ContentReferenceValue .
ObjectReferenceValue  ::= ":ObjectRef" ObjectReference .
ContentReferenceValue ::= ":ContentRef" ContentReference .
```

B.4.15 BooleanVariable Class

```
BooleanVariableClass  ::= "{:BooleanVar" Variable "}" .
```

B.4.16 IntegerVariable Class

```
IntegerVariableClass  ::= "{:IntegerVar" Variable "}" .
```

B.4.17 OctetStringVariable Class

```
OctetStringVariableClass      ::= "{:OStringVar" Variable "}" .
```

B.4.18 ObjectRefVariable Class

```
ObjectRefVariableClass ::= "{:ObjectRefVar" Variable "}" .
```

B.4.19 ContentRefVariable Class

```
ContentRefVariableClass       ::= "{:ContentRefVar" Variable "}" .
```

B.4.20 Presentable Class

```
Presentable           ::= Ingredient .
```

B.4.21 TokenManager Class

```
TokenManager          ::= [MovementTable] .
MovementTable         ::= ":MovementTable" "(" Movement+ ")" .
Movement              ::= "(" TargetElement+ ")" .
TargetElement         ::= INTEGER .
```

B.4.22 TokenGroup Class

```
TokenGroupClass       ::= "{:TokenGroup" TokenGroupBody "}" .
TokenGroupBody        ::= Presentable TokenManager TokenGroupItems
                          [NoTokenActionSlots] .
TokenGroupItems       ::= ":TokenGroupItems" "(" TokenGroupItem+ ")" .
TokenGroupItem        ::= "(" AVisible [ActionSlots] ")" .
AVisible              ::= ObjectReference .
ActionSlots           ::= ":ActionSlots" "(" ActionSlot+ ")" .
ActionSlot            ::= ActionClass | Null .
NoTokenActionSlots    ::= ":NoTokenActionSlots" "(" ActionSlot+ ")" .
```

B.4.23 ListGroup Class

```
ListGroupClass        ::= "{:ListGroup" TokenGroupBody
                          Positions [WrapAround]
                          [MultipleSelection] "}" .
Positions             ::= ":Positions" "(" Position+ ")" .
Position              ::= "(" XYPosition ")" .
WrapAround            ::= ":WrapAround" BOOLEAN .
MultipleSelection     ::= ":MultipleSelection" BOOLEAN .
```

B.4.24 Visible Class

```
Visible               ::= Presentable OriginalBoxSize
                          [OriginalPosition] [OriginalPaletteRef] .
OriginalBoxSize       ::= ":OrigBoxSize" BoxSize .
BoxSize               ::= XLength YLength .
XLength               ::= INTEGER .
YLength               ::= INTEGER .
OriginalPosition      ::= ":OrigPosition" XYPosition .
OriginalPaletteRef    ::= ":OrigPaletteRef" ObjectReference .
```

B.4.25 Bitmap Class

```
BitmapClass           ::= "{:Bitmap" Visible [Tiling]
                          [OriginalTransparency] "}" .
Tiling                ::= ":Tiling" BOOLEAN .
OriginalTransparency  ::= ":OrigTransparency" INTEGER .
```

B.4.26 LineArt Class

```
LineArtClass          ::= "{:LineArt" LineArtBody "}" .
LineArtBody           ::= Visible [BorderedBoundingBox]
                          [OriginalLineWidth]
                          [OriginalLineStyle]
                          [OriginalRefLineColour]
                          [OriginalRefFillColour] .
BorderedBoundingBox   ::= ":BBBox" BOOLEAN .
OriginalLineWidth     ::= ":OrigLineWidth" INTEGER .
OriginalLineStyle     ::= ":OrigLineStyle" INTEGER .
OriginalRefLineColour ::= ":OrigRefLineColour" Colour .
OriginalRefFillColour ::= ":OrigRefFillColour" Colour .
```

B.4.27 Rectangle Class

```
RectangleClass        ::= "{:Rectangle" LineArtBody "}" .
```

B.4.28 DynamicLineArt Class

```
DynamicLineArtClass   ::= "{:DynamicLineArt" LineArtBody "}" .
```

B.4.29 Text Class

```
TextClass             ::= "{:Text" TextBody "}" .
TextBody              ::= Visible [OriginalFont] [FontAttributes]
                          [TextColour] [BackgroundColour]
                          [CharacterSet]
                          [HorizontalJustification]
                          [VerticalJustification]
                          [LineOrientation] [StartCorner]
                          [TextWrapping] .
OriginalFont   ::= ":OrigFont" FontBody .
HorizontalJustification ::= ":HJustification" JustificationEnum .
JustificationEnum     ::= "start" | "end" | "centre" | "justified" .
VerticalJustification ::= ":VJustification" JustificationEnum .
LineOrientation       ::= ":LineOrientation" LineOrientationEnum .
LineOrientationEnum   ::= "vertical" | "horizontal" .
StartCorner           ::= ":StartCorner" StartCornerEnum .
StartCornerEnum       ::= "upper-left" | "upper-right"
                        | "lower-left" | "lower-right" .
TextWrapping          ::= ":TextWrapping" BOOLEAN .
```

B.4.30 Stream Class

```
StreamClass           ::= "{:Stream" Presentable Multiplex
                          [Storage] [Looping] "}" .
Multiplex             ::= ":Multiplex" "(" StreamComponent+ ")" .
StreamComponent       ::= AudioClass | VideoClass | RTGraphicsClass .
Storage               ::= ":Storage" StorageEnum .
StorageEnum           ::= "memory" | "stream" .
Looping               ::= ":Looping" INTEGER .
```

B.4.31 Audio Class

```
AudioClass            ::= "{:Audio" Presentable ComponentTag
                          [OriginalVolume] "}" .
ComponentTag          ::= ":ComponentTag" INTEGER .
OriginalVolume        ::= ":OrigVolume" INTEGER .
```

B.4.32 Video Class

```
VideoClass            ::= "{:Video" Visible ComponentTag
                          [Termination] "}" .
```

```
Termination             ::= ":Termination" TerminationEnum .
TerminationEnum         ::= "freeze" | "disappear" .
```

B.4.33 RTGraphics Class

```
RTGraphicsClass         ::= "{:RTGraphics" Visible ComponentTag
                            [Termination] "}" .
```

B.4.34 Interactible Class

```
Interactible            ::= [EngineResp] [HighlightRefColour] .
EngineResp              ::= ":EngineResp" BOOLEAN .
```

B.4.35 Slider Class

```
SliderClass             ::= "{:Slider" Visible Interactible
                            Orientation MaxValue [MinValue]
                            [InitialValue] [InitialPortion]
                            [StepSize] [SliderStyle]
                            [SliderRefColour] "}" .
Orientation             ::= ":Orientation" OrientationEnum .
OrientationEnum         ::= "left" | "right" | "up" | "down" .
MaxValue                ::= ":MaxValue" INTEGER .
MinValue                ::= ":MinValue" INTEGER .
InitialValue            ::= ":InitialValue" INTEGER .
InitialPortion          ::= ":InitialPortion" INTEGER .
StepSize                ::= ":StepSize" INTEGER .
SliderStyle             ::= ":SliderStyle" SliderStyleEnum .
SliderStyleEnum         ::= "normal" | "thermometer" | "proportional" .
```

B.4.36 EntryField Class

```
EntryFieldClass         ::= "{:EntryField" TextBody Interactible
                            [InputType] [CharList]
                            [ObscuredInput] [MaxLength] "}" .
InputType               ::= ":InputType" InputTypeEnum .
InputTypeEnum           ::= "alpha" | "numeric" | "any" | "listed" .
CharList                ::= ":CharList" OctetString .
ObscuredInput           ::= ":ObscuredInput" BOOLEAN .
MaxLength               ::= ":MaxLength" INTEGER .
```

B.4.37 HyperText Class

```
HyperTextClass          ::= "{:HyperText" TextBody Interactible
                            "}" .
```

B.4.38 Button Class

```
Button                  ::= Visible Interactible [ButtonRefColour] .
```

B.4.39 Hotspot Class

```
HotspotClass            ::= "{:Hotspot" Button "}" .
```

B.4.40 PushButton Class

```
PushButtonClass         ::= "{:PushButton" PushButtonBody "}" .
PushButtonBody          ::= Button [OriginalLabel] [CharacterSet] .
OriginalLabel           ::= ":OrigLabel" OctetString .
```

B.4.41 SwitchButton Class

```
SwitchButtonClass       ::= "{:SwitchButton" PushButtonBody
                            ButtonStyle "}" .
ButtonStyle             ::= ":ButtonStyle" ButtonStyleEnum .
ButtonStyleEnum         ::= "pushbutton" | "radiobutton" | "checkbox" .
```

B.4.42 Applet Class

```
AppletClass           ::= "{" ":Applet" Program Interactible
                          OriginalBoxSize [OriginalPosition] "}".
```

B.4.43 Action Class

```
ActionClass           ::= "(" ElementaryAction+ ")" .
ElementaryAction      ::= Activate
                        | Add
                        | AddItem
                        | Append
                        | BringToFront
                        | Call
                        | CallActionSlot
                        | Clear
                        | Clone
                        | CloseConnection
                        | Deactivate
                        | DelItem
                        | Deselect
                        | DeselectItem
                        | Divide
                        | DrawArc
                        | DrawLine
                        | DrawOval
                        | DrawPolygon
                        | DrawPolyline
                        | DrawRectangle
                        | DrawSector
                        | Fork
                        | GetAvailabilityStatus
                        | GetBoxSize
                        | GetCellItem
                        | GetCursorPosition
                        | GetEngineSupport
                        | GetEntryPoint
                        | GetFillColour
                        | GetFirstItem
                        | GetHighlightStatus
                        | GetInteractionStatus
                        | GetItemStatus
                        | GetLabel
                        | GetLastAnchorFired
                        | GetLineColour
                        | GetLineStyle
                        | GetLineWidth
                        | GetListItem
                        | GetListSize
                        | GetOverwriteMode
                        | GetPortion
                        | GetPosition
                        | GetRunningStatus
                        | GetSelectionStatus
                        | GetSliderValue
                        | GetTextContent
                        | GetTextData
                        | GetTokenPosition
                        | GetVolume
                        | Invoke
                        | Launch
                        | LockScreen
                        | Modulo
                        | Move
                        | MoveTo
                        | Multiply
                        | OpenConnection
                        | Preload
                        | PutBefore
                        | PutBehind
                        | Quit
                        | ReadPersistent
                        | Run
                        | ScaleBitmap
                        | ScaleVideo
                        | ScrollItems
                        | Select
```

```
                        | SelectItem
                        | SendEvent
                        | SendToBack
                        | SetBoxSize
                        | SetCachePriority
                        | SetCounterEndPosition
                        | SetCounterPosition
                        | SetCounterTrigger
                        | SetCursorPosition
                        | SetCursorShape
                        | SetData
                        | SetEntryPoint
                        | SetFillColour
                        | SetFirstItem
                        | SetFontRef
                        | SetHighlightStatus
                        | SetInteractionStatus
                        | SetLabel
                        | SetLineColour
                        | SetLineStyle
                        | SetLineWidth
                        | SetOverwriteMode
                        | SetPaletteRef
                        | SetPortion
                        | SetPosition
                        | SetSliderValue
                        | SetSpeed
                        | SetTimer
                        | SetTransparency
                        | SetVariable
                        | SetVolume
                        | Spawn
                        | Step
                        | Stop
                        | StorePersistent
                        | Subtract
                        | TestVariable
                        | Toggle
                        | ToggleItem
                        | TransitionTo
                        | Unload
                        | UnlockScreen.

Activate              ::= ":Activate" "(" Target ")" .
Add                   ::= ":Add" "(" Target Value ")" .
AddItem               ::= ":AddItem" "(" Target ItemIndex
                          VisibleReference ")" .
Append                ::= ":Append" "(" Target AppendValue ")" .
BringToFront          ::= ":BringToFront" "(" Target ")" .
Call                  ::= ":Call" "(" Target CallSucceeded
                          [Parameters] ")" .
CallActionSlot        ::= ":CallActionSlot" "(" Target Index ")" .
Clear                 ::= ":Clear" "(" Target ")" .
Clone                 ::= ":Clone" "(" Target CloneRefVar ")" .
CloseConnection       ::= ":CloseConnection" "(" Target
                          ConnectionTag ")" .
Deactivate            ::= ":Deactivate" "(" Target ")" .
DelItem               ::= ":DelItem" "(" Target VisibleReference
                          ")" .
Deselect              ::= ":Deselect" "(" Target ")" .
DeselectItem          ::= ":DeselectItem" "(" Target ItemIndex ")" .
Divide                ::= ":Divide" "(" Target Value ")" .
DrawArc               ::= ":DrawArc" "(" Target X Y EllipseWidth
                          EllipseHeight StartAngle ArcAngle ")" .
DrawLine              ::= ":DrawLine" "(" Target X1 Y1 X2 Y2 ")" .
DrawOval              ::= ":DrawOval" "(" Target X Y EllipseWidth
                          EllipseHeight ")" .
DrawPolygon           ::= ":DrawPolygon" "(" Target PointList ")" .
DrawPolyline          ::= ":DrawPolyline" "(" Target PointList ")" .
DrawRectangle         ::= ":DrawRectangle" "(" Target X1 Y1 X2 Y2
                          ")" .
DrawSector            ::= ":DrawSector" "(" Target X Y EllipseWidth
                          EllipseHeight StartAngle ArcAngle ")" .
Fork                  ::= ":Fork" "(" Target ForkSucceeded
                          [Parameters] ")" .
GetAvailabilityStatus ::= ":GetAvailabilityStatus" "(" Target
                          AvailabilityStatusVar ")" .
GetBoxSize            ::= ":GetBoxSize" "(" Target XBoxSizeVar
                          YBoxSizeVar ")" .
```

```
GetCellItem             ::= ":GetCellItem" "(" Target CellIndex
                            ItemRefVar ")" .
GetCursorPosition       ::= ":GetCursorPosition" "(" Target XOut YOut
                            ")" .
GetEngineSupport        ::= ":GetEngineSupport" "(" Target Feature
                            Answer ")" .
GetEntryPoint           ::= ":GetEntryPoint" "(" Target EntryPointVar
                            ")" .
GetFillColour           ::= ":GetFillColour" "(" Target FillColourVar
                            ")" .
GetFirstItem            ::= ":GetFirstItem" "(" Target FirstItemVar
                            ")" .
GetHighlightStatus      ::= ":GetHighlightStatus" "(" Target
                            HighlightStatusVar ")" .
GetInteractionStatus    ::= ":GetInteractionStatus" "(" Target
                            InteractionStatusVar ")" .
GetItemStatus           ::= ":GetItemStatus" "(" Target
                            ItemIndex ItemStatusVar ")" .
GetLabel                ::= ":GetLabel" "(" Target LabelVar ")" .
GetLastAnchorFired      ::= ":GetLastAnchorFired" "(" Target
                            LastAnchorFiredVar ")" .
GetLineColour           ::= ":GetLineColour" "(" Target LineColourVar
                            ")" .
GetLineStyle            ::= ":GetLineStyle" "(" Target LineStyleVar
                            ")" .
GetLineWidth            ::= ":GetLineWidth" "(" Target LineWidthVar
                            ")" .
GetListItem             ::= ":GetListItem" "(" Target ItemIndex
                            ItemRefVar ")" .
GetListSize             ::= ":GetListSize" "(" Target SizeVar ")" .
GetOverwriteMode        ::= ":GetOverwriteMode" "(" Target
                            OverwriteModeVar ")" .
GetPortion              ::= ":GetPortion" "(" Target PortionVar ")" .
GetPosition             ::= ":GetPosition" "(" Target XPositionVar
                            YPositionVar ")" .
GetRunningStatus        ::= ":GetRunningStatus" "(" Target
                            RunningStatusVar ")" .
GetSelectionStatus      ::= ":GetSelectionStatus" "(" Target
                            SelectionStatusVar ")" .
GetSliderValue          ::= ":GetSliderValue" "(" Target
                            SliderValueVar ")" .
GetTextContent          ::= ":GetTextContent" "(" Target
                            TextContentVar ")" .
GetTextData             ::= ":GetTextData" "(" Target TextDataVar ")" .
GetTokenPosition        ::= ":GetTokenPosition" "(" Target
                            TokenPositionVar ")" .
GetVolume               ::= ":GetVolume" "(" Target VolumeVar ")" .
Invoke                  ::= ":Invoke" "(" Target InvokeSucceedeed Method [Parameters] ")".
Launch                  ::= ":Launch" "(" Target ")" .
LockScreen              ::= ":LockScreen" "(" Target ")" .
Modulo                  ::= ":Modulo" "(" Target Value ")" .
Move                    ::= ":Move" "(" Target MovementIdentifier ")" .
MoveTo                  ::= ":MoveTo" "(" Target Index ")" .
Multiply                ::= ":Multiply" "(" Target Value ")" .
OpenConnection          ::= ":OpenConnection" "(" Target OpenSucceeded
                            Protocol Address ConnectionTag ")" .
Preload                 ::= ":Preload" "(" Target ")" .
PutBefore               ::= ":PutBefore" "(" Target ReferenceVisible
                            ")" .
PutBehind               ::= ":PutBehind" "(" Target ReferenceVisible
                            ")" .
Quit                    ::= ":Quit" "(" Target ")" .
ReadPersistent          ::= ":ReadPersistent" "(" Target ReadSucceeded
                            OutVariables InFileName ")" .
Run                     ::= ":Run" "(" Target ")" .
ScaleBitmap             ::= ":ScaleBitmap" "(" Target XScale YScale
                            ")" .
ScaleVideo              ::= ":ScaleVideo" "(" Target XScale YScale ")" .
ScrollItems             ::= ":ScrollItems" "(" Target ItemsToScroll ")" .
Select                  ::= ":Select" "(" Target ")" .
SelectItem              ::= ":SelectItem" "(" Target ItemIndex ")" .
SendEvent               ::= ":SendEvent" "(" Target EmulatedEventSource
                            EmulatedEventType [EmulatedEventData]
                            ")" .
SendToBack              ::= ":SendToBack" "(" Target ")" .
SetBoxSize              ::= ":SetBoxSize" "(" Target XNewBoxSize
                            YNewBoxSize ")" .
SetCachePriority        ::= ":SetCachePriority" "(" Target
                            NewCachePriority ")" .
```

```
SetCounterEndPosition  ::= ":SetCounterEndPosition" "(" Target
                           NewCounterEndPosition ")" .
SetCounterPosition     ::= ":SetCounterPosition" "(" Target
                           NewCounterPosition ")" .
SetCounterTrigger      ::= ":SetCounterTrigger" "(" Target
                           TriggerIdentifier [NewCounterValue]
                           ")" .
SetCursorPosition      ::= ":SetCursorPosition" "(" Target XCursor
                           YCursor ")" .
SetCursorShape         ::= ":SetCursorShape" "(" Target
                           [NewCursorShape] ")" .
SetData                ::= ":SetData" "(" Target NewContent ")" .
SetEntryPoint          ::= ":SetEntryPoint" "(" Target NewEntryPoint
                           ")" .
SetFillColour          ::= ":SetFillColour" "(" Target [NewColour]
                           ")" .
SetFirstItem           ::= ":SetFirstItem" "(" Target NewFirstItem
                           ")" .
SetFontRef             ::= ":SetFontRef" "(" Target NewFont ")" .
SetHighlightStatus     ::= ":SetHighlightStatus" "(" Target
                           NewHighlightStatus ")" .
SetInteractionStatus   ::= ":SetInteractionStatus" "(" Target
                           NewInteractionStatus ")" .
SetLabel               ::= ":SetLabel" "(" Target NewLabel ")" .
SetLineColour          ::= ":SetLineColour" "(" Target NewColour ")" .
SetLineStyle           ::= ":SetLineStyle" "(" Target NewLineStyle
                           ")" .
SetLineWidth           ::= ":SetLineWidth" "(" Target NewLineWidth
                           ")" .
SetOverwriteMode       ::= ":SetOverwriteMode" "(" Target
                           NewOverwriteMode ")" .
SetPaletteRef          ::= ":SetPaletteRef" "(" Target NewPaletteRef
                           ")" .
SetPortion             ::= ":SetPortion" "(" Target NewPortion ")" .
SetPosition            ::= ":SetPosition" "(" Target NewXPosition
                           NewYPosition ")" .
SetSliderValue         ::= ":SetSliderValue" "(" Target
                           NewSliderValue ")" .
SetSpeed               ::= ":SetSpeed" "(" Target NewSpeed ")" .
SetTimer               ::= ":SetTimer" "(" Target TimerID
                           [TimerValue] [AbsoluteTime] ")" .
SetTransparency        ::= ":SetTransparency" "(" Target
                           NewTransparency ")" .
SetVariable            ::= ":SetVariable" "(" Target
                           NewVariableValue ")" .
SetVolume              ::= ":SetVolume" "(" Target NewVolume ")" .
Spawn                  ::= ":Spawn" "(" Target ")" .
Stop                   ::= ":Stop" "(" Target ")" .
Step                   ::= ":Step" "(" Target NbOfSteps ")" .
StorePersistent        ::= ":StorePersistent" "(" Target
                           StoreSucceeded InVariables OutFileName
                           ")" .
Subtract               ::= ":Subtract" "(" Target Value ")" .
TestVariable           ::= ":TestVariable" "(" Target Operator
                           ComparisonValue ")" .
Toggle                 ::= ":Toggle" "(" Target ")" .
ToggleItem             ::= ":ToggleItem" "(" Target ItemIndex ")" .
TransitionTo           ::= ":TransitionTo" "(" Target [ConnectionTag]
                           [TransitionEffect] ")" .
Unload                 ::= ":Unload" "(" Target ")" .
UnlockScreen           ::= ":UnlockScreen" "(" Target ")" .

AbsoluteTime           ::= ":AbsoluteTime" GenericBoolean .
Address                ::= GenericOctetString .
Answer                 ::= ObjectReference .
AppendValue            ::= GenericOctetString .
ArcAngle               ::= GenericInteger .
AvailabilityStatusVar  ::= ObjectReference .
CallSucceeded          ::= ObjectReference .
CellIndex              ::= GenericInteger .
CloneRefVar            ::= ObjectReference .
ComparisonValue        ::= NewGenericBoolean | NewGenericInteger
                           | NewGenericOctetString
                           | NewGenericObjectReference
                           | NewGenericContentReference .
ConnectionTag          ::= ":ConnectionTag" GenericInteger .
Denominator            ::= GenericInteger .
EllipseHeight          ::= GenericInteger .
EllipseWidth           ::= GenericInteger .
```

```
EmulatedEventData      ::= NewGenericBoolean | NewGenericInteger
                           | NewGenericOctetString .
EmulatedEventSource    ::= GenericObjectReference .
EmulatedEventType      ::= EventTypeEnum .
EntryPointVar          ::= ObjectReference .
ForkSucceeded          ::= ObjectReference .
Feature                ::= GenericOctetString .
FillColourVar          ::= ObjectReference .
FirstItemVar           ::= ObjectReference .
HighlightStatusVar     ::= ObjectReference .
Index                  ::= GenericInteger .
InFileName             ::= GenericOctetString .
InteractionStatusVar   ::= ObjectReference .
InVariables            ::= "(" ObjectReference+ ")" .
InvokeSucceeded        ::= ObjectReference.
ItemIndex              ::= GenericInteger .
ItemRefVar             ::= ObjectReference .
ItemStatusVar          ::= ObjectReference .
ItemsToScroll          ::= GenericInteger .
LabelVar               ::= ObjectReference .
LastAnchorFiredVar     ::= ObjectReference .
LineColourVar          ::= ObjectReference .
LineStyleVar           ::= ObjectReference .
LineWidthVar           ::= ObjectReference .
Method                 ::= GenericOctetString.
MovementIdentifier     ::= GenericInteger .
NbOfSteps              ::= GenericInteger .
NewAbsoluteColour      ::= ":NewAbsoluteColour" GenericOctetString .
NewCachePriority       ::= GenericInteger .
NewColour              ::= NewColourIndex | NewAbsoluteColour .
NewColourIndex         ::= ":NewColourIndex" GenericInteger .
NewContent             ::= NewIncludedContent | NewReferencedContent .
NewContentCachePriority ::= ":NewCCPriority" GenericInteger .
NewCounterEndPosition  ::= GenericInteger .
NewCounterPosition     ::= GenericInteger .
NewContentSize         ::= ":NewContentSize" GenericInteger .
NewCounterValue        ::= GenericInteger .
NewCursorShape         ::= GenericObjectReference .
NewEntryPoint          ::= GenericInteger .
NewFirstItem           ::= GenericInteger .
NewFont                ::= NewFontName | NewFontReference .
NewFontName            ::= NewGenericOctetString .
NewFontReference       ::= NewGenericObjectReference .
NewGenericBoolean      ::= ":GBoolean" GenericBoolean .
NewGenericInteger      ::= ":GInteger" GenericInteger .
NewGenericOctetString  ::= ":GOctetString" GenericOctetString .
NewGenericObjectReference ::= ":GObjectRef" GenericObjectReference .
NewGenericContentReference ::= ":GContentRef" GenericContentReference .
NewHighlightStatus     ::= GenericBoolean .
NewIncludedContent     ::= GenericOctetString .
NewInteractionStatus   ::= GenericBoolean .
NewLabel               ::= GenericOctetString .
NewLineStyle           ::= GenericInteger .
NewLineWidth           ::= GenericInteger .
NewOverwriteMode       ::= GenericBoolean .
NewPaletteRef          ::= GenericObjectReference .
NewPortion             ::= GenericInteger .
NewReferencedContent   ::= ":NewRefContent" "(" GenericContentReference
                           [NewContentSize]
                           [NewContentCachePriority] ")" .
NewSliderValue         ::= GenericInteger .
NewSpeed               ::= Rational .
NewTransparency        ::= GenericInteger .
NewVariableValue       ::= NewGenericInteger | NewGenericBoolean
                           | NewGenericOctetString
                           | NewGenericObjectReference
                           | NewGenericContentReference .
NewVolume              ::= GenericInteger .
NewXPosition           ::= GenericInteger .
NewYPosition           ::= GenericInteger .
Numerator              ::= GenericInteger .
OpenSucceeded          ::= ObjectReference .
Operator               ::= GenericInteger .
OutFileName            ::= GenericOctetString .
OutVariables           ::= "(" ObjectReference+ ")" .
OverwriteModeVar       ::= ObjectReference .
Parameter              ::= NewGenericBoolean | NewGenericInteger
                           | NewGenericOctetString
                           | NewGenericObjectReference
```

```
                        | NewGenericContentReference .
Parameters              ::= Parameter+ .
Point                   ::= "(" X Y ")" .
PointList               ::= "(" Point+ ")" .
PortionVar              ::= ObjectReference .
Protocol                ::= GenericOctetString .
Rational                ::= Numerator [Denominator] .
ReadSucceeded           ::= ObjectReference .
ReferenceVisible        ::= GenericObjectReference .
RunningStatusVar        ::= ObjectReference .
SelectionStatusVar      ::= ObjectReference .
SizeVar                 ::= ObjectReference .
SliderValueVar          ::= ObjectReference .
StartAngle              ::= GenericInteger .
StoreSucceeded          ::= ObjectReference .
Target                  ::= GenericObjectReference .
TextContentVar          ::= ObjectReference .
TextDataVar             ::= ObjectReference .
TimerID                 ::= GenericInteger .
TimerValue              ::= GenericInteger .
TokenPositionVar        ::= ObjectReference .
TransitionEffect        ::= GenericInteger .
TriggerIdentifier       ::= GenericInteger .
Value                   ::= GenericInteger .
VisibleReference        ::= GenericObjectReference .
VolumeVar               ::= ObjectReference .
X                       ::= GenericInteger .
X1                      ::= GenericInteger .
X2                      ::= GenericInteger .
XBoxSizeVar             ::= ObjectReference .
XCursor                 ::= GenericInteger .
XNewBoxSize             ::= GenericInteger .
XOut                    ::= ObjectReference .
XPositionVar            ::= ObjectReference .
XScale                  ::= GenericInteger .
Y                       ::= GenericInteger .
Y1                      ::= GenericInteger .
Y2                      ::= GenericInteger .
YBoxSizeVar             ::= ObjectReference .
YCursor                 ::= GenericInteger .
YNewBoxSize             ::= GenericInteger .
YOut                    ::= ObjectReference .
YPositionVar            ::= ObjectReference .
YScale                  ::= GenericInteger .
```

B.4.44 Miscellaneous data types

```
ObjectReference         ::= ExternalReference | InternalReference .
ExternalReference       ::= "(" GroupIdentifier ObjectNumber ")" .
InternalReference       ::= ObjectNumber .
GroupIdentifier         ::= OctetString .
ObjectNumber            ::= INTEGER .

ContentReference        ::= OctetString .

GenericObjectReference ::= DirectReference | IndirectReference .
DirectReference         ::= ObjectReference .
IndirectReference       ::= ":IndirectRef" ObjectReference .

GenericContentReference ::= ContentReference | IndirectReference .

GenericInteger          ::= INTEGER | IndirectReference .

GenericBoolean          ::= BOOLEAN | IndirectReference .

GenericOctetString      ::= OctetString | IndirectReference .

OctetString             ::= STRING | QPRINTABLE | BASE64 .

Colour                  ::= ColourIndex | AbsoluteColour .
ColourIndex             ::= INTEGER .
AbsoluteColour          ::= OctetString .

XYPosition              ::= XPosition YPosition .
XPosition               ::= INTEGER .
YPosition               ::= INTEGER .
```

Annex C
(normative)

MHEG-5 API

The MHEG-5 API is defined as a Java specification. It consists of one Java package, called `iso.mheg5`.

NOTE 1 The following specification was developed jointly with ETSI, and is technically aligned with ETS 300 777-1.

Within the iso.mheg5 package,

- any public class that represents an MHEG-5 object class shall have a protected constructor without parameter;
- any public method of a public class that represents an MHEG-5 object class shall be declared to throw MhegException.

NOTE 2 These protected constructors and exceptions are not described in the text. Compilable Java code can be obtained by clustering the following declarations (Courier lines) into class files, adding protected constructors and declaring throwable exceptions as specified above and appending bodies to class method declarations and semicolons to other declarations.

The complete grade consists of all classes listed in this Annex. The reduced grade consists of the classes listed in Clauses C.1 to C.3, C.5 to C.10 and C.19 to C.24.

C.1 ObjectReference

```
public class ObjectReference
```

Fields

```
public OctetString groupIdentifier

public int objectNumber
```

Constructors

```
public ObjectReference()
public ObjectReference(int objectNumber)
public ObjectReference(OctetString groupIdentifier,
                       int objectNumber)
```

The first two constructors initialise the groupIdentifier attribute (not provided) to a zero-length octetstring.

NOTE ObjectReference has public fields and is therefore mutable. This allows to spare ObjectReference objects, which are often used as transitory variables to access objects, for later re-use. Programmers should be aware that the value of an ObjectReference object may be modified by JVM code to which the object is passed.

C.2 ContentReference

```
public class ContentReference
```

Constructors

```
public ContentReference()
public ContentReference(OctetString reference)
```

C.3 OctetString

```
public class OctetString
```

Constructors

```
public OctetString(String value)
public OctetString(byte[] value)
```

Methods

`byte` **`byteAt`**`(int index)`
Returns the value of the byte at position index in the octet string.

`int` **`length`** `()`
Returns the number of bytes of the octet string.
The returned value shall be positive.

`byte[]` **`toByteArray`**`()`
Returns the octet string as a byte array.

`String` **`toString`**`()`
Returns a character string corresponding to the octet string using the applicable MHEG-5 text format.

C.4 Colour

```
public class Colour
```

Constructors

```
public Colour(int colourIndex)
public Colour(OctetString absoluteColour)
```

Methods

`boolean` **`isAbsolute`**`()`
Returns TRUE if the Colour object has been created as an absolute colour, FALSE if it has been created as a colour index.

`int` **`getIndexColour`**`()`
Returns the colour index.
If the object has been created as an absolute colour, an exception shall be raised.

`OctetString` **`getAbsoluteColour`**`()`
Returns the colour name.
If the object has been created as a colour index, an exception shall be raised.

C.5 MhegException

```
public class MhegException extends java.lang.Exception
```

Fields

```
public short parameterRank
```

```
public short exceptionCode
```
The legal values for exceptionCode are TARGET_NOT_AVAILABLE, INVALID_TARGET, ILLEGAL_PARAMETER, OPTION_NOT_SUPPORTED.

Constants

Legal values for exceptionCode:

`public static final short TARGET_NOT_AVAILABLE = 1`
This exception shall be raised when a method is targeted at a non-available MHEG-5 object.

`public static final short INVALID_TARGET = 2`
This exception shall be raised when a method is targeted at an MHEG-5 object whose reference is invalid.

`public static final short ILLEGAL_PARAMETER = 3`
This exception shall be raised when a method is invoked with parameter values that are illegal.

`public static final short OPTION_NOT_SUPPORTED = 4`
This exception shall be raised when a method needs to call upon optional features not supported by the engine or the application domain.

Constructors

`public MhegException(short reason)`
Constructs an MhegException with exceptionCode identified by the reason parameter.

```
public MhegException(short reason,
                     short position)
```
Constructs an MhegException with exceptionCode identified by the reason parameter and parameterRank by the position parameter.

NOTE As with ObjectReference, MHEGException is a mutable object which is often used for transitory purposes. Programmers should remain aware that the fields of an MHEGException object may be modified by JVM code to which the object is passed.

C.6 Root

`abstract public class Root`

Methods

`public static ObjectReference getOwnId()`
Returns the MHEG-5 identification of the calling InterchangedProgram or Applet.

`public static Root getObject(ObjectReference mheg5ObjectReference)`
Returns the reference of the Java object associated with the MHEG-5 object whose identification is mheg5ObjectReference.
If the Java «proxy» object does not exist, creates it first.

`public ObjectReference getReference()`
Creates a reference to the object and returns this reference.

`public boolean getAvailabilityStatus()`
Retrieves the value of the AvailabilityStatus internal attribute.

`public boolean getRunningStatus()`
Retrieves the value of the RunningStatus internal attribute.

C.7 Group

`abstract public class Group extends Root`

Methods

`public void setCachePriority(short cachePriority)`
Triggers execution of the SetCachePriority elementary action.
The value of the cachePriority parameter shall be within the range [0,255].

```
public short getCachePriority()
```
Retrieves the value of the GroupCachePriority internal attribute.

C.8 Application

```
public class Application extends Group
```

Methods

```
public void lockScreen()
```
Triggers execution of the LockScreen elementary action.

```
public void unlockScreen()
```
Triggers execution of the UnlockScreen elementary action.

```
public int getLockCount()
```
Retrieves the value of the LockCount internal attribute.
The returned value shall be positive.

```
public boolean getEngineSupport(OctetString feature)
```
Triggers execution of the GetEngineSupport elementary action.

C.9 Scene

```
public class Scene extends Group
```

Constants

Legal values for eventType:

```
public static final byte IS_AVAILABLE = 1
public static final byte CONTENT_AVAILABLE = 2
public static final byte IS_DELETED = 3
public static final byte IS_RUNNING = 4
public static final byte IS_STOPPED = 5
public static final byte USER_INPUT = 6
public static final byte ANCHOR_FIRED = 7
public static final byte TIMER_FIRED = 8
public static final byte ASYNCH_STOPPED = 9
public static final byte INTERACTION_COMPLETED = 10
public static final byte TOKEN_MOVED_FROM = 11
public static final byte TOKEN_MOVED_TO = 12
public static final byte STREAM_EVENT = 13
public static final byte STREAM_PLAYING = 14
public static final byte STREAM_STOPPED = 15
public static final byte COUNTER_TRIGGER = 16
public static final byte HIGHLIGHT_ON = 17
public static final byte HIGHLIGHT_OFF = 18
public static final byte CURSOR_ENTER = 19
public static final byte CURSOR_LEAVE = 20
public static final byte IS_SELECTED = 21
public static final byte IS_DESELECTED = 22
public static final byte TEST_EVENT = 23
public static final byte FIRST_ITEM_PRESENTED = 24
public static final byte LAST_ITEM_PRESENTED = 25
public static final byte HEAD_ITEMS = 26
public static final byte TAIL_ITEMS = 27
public static final byte ITEM_SELECTED = 28
public static final byte ITEM_DESELECTED = 29
public static final byte ENTRY_FIELD_FULL = 30
public static final byte ENGINE_EVENT = 31
```

Methods

```
public void setTimer(int timerId)
public void setTimer(int timerId,
                     int timerValue)
public void setTimer(int timerId,
                     int timerValue,
                     boolean absoluteTime)
```
Triggers execution of the SetTimer elementary action.

```
public int getTimerPosition(int timerId)
```
Retrieves the value of the TimerPosition field of the timer identified by timerId in the Timers internal attribute.
If timerId does not refer to a valid timer, an exception shall be raised.

```
public boolean getAbsoluteTime(int timerId)
```
Retrieves the value of the AbsoluteTime field of the timer identified by timerId in the Timers internal attribute.
If timerId does not refer to a valid timer, an exception shall be raised.

```
public void sendEvent(ObjectReference eventSource,
                      byte eventType)
public void sendEvent(ObjectReference eventSource,
                      byte eventType,
                      boolean eventData)
public void sendEvent(ObjectReference eventSource,
                      byte eventType,
                      int eventData)
public void sendEvent(ObjectReference eventSource,
                      byte eventType,
                      OctetString eventData)
```
Triggers execution of the SendEvent elementary action.

```
public void setCursorShape()
public void setCursorShape(ObjectReference cursorShape)
```
Triggers execution of the SetCursorShape elementary action.

```
public ObjectReference getCursorShape()
```
Retrieves the object reference to the CursorShape object currently attached to the scene.
A returned null object reference indicates that the scene currently has no associated moving cursor.

```
public void setCursorPosition(short xCursor,
                              short yCursor)
```
Triggers execution of the SetCursorPosition elementary action.

```
public short[] getCursorPosition()
```
Retrieves the value of the CursorPosition internal attribute.
The returned array shall have two elements, the first is the XCursor, the second is the YCursor.

C.10 Ingredient

```
abstract public class Ingredient extends Root
```

Methods

```
public void setContentData(OctetString includedContent)
```
Triggers execution of the SetData elementary action, for an included content.
For the purpose of consistency with the semantics of getContentData, the method that applies SetData to an included content is called setContentData.
The effect is not specified if the Content attribute is referenced. It may be defined by an application domain.

```
public void setContentReference(ContentReference referencedContent)
public void setContentReference(ContentReference referencedContent,
                                int contentSize)
```

```
public void setContentReference(ContentReference referencedContent,
                                short contentCachePriority)
public void setContentReference(ContentReference referencedContent,
                                int contentSize,
                                short contentCachePriority)
```

Triggers execution of the SetData elementary action, for a referenced content.
If provided, the value of the contentCachePriority parameter shall be within the range [0,255].
For the purpose of consistency with the semantics of getContentReference, the methods that apply SetData to a referenced content are called setContentReference.
If the Content attribute is included, an exception shall be raised.

```
public OctetString getContentData()
```

Retrieves the actual content of the Ingredient.
If the content is included, the returned value shall be the value of the Content internal attribute.
If the content is referenced, the returned value shall be the byte array representation of the value of the external data referenced by the Content internal attribute.
Note that if the Ingredient is a Text object, this method retrieves the value of the TextData internal attribute.

```
public ContentReference getContentReference()
```

Retrieves the value of the ContentReference field of the Content internal attribute.
If the content is included, a null object reference shall be returned.

```
public int getContentSize()
```

Retrieves the value of the ContentSize field of the Content internal attribute.
If the content is included, an exception shall be raised.
If the field is not provided, the returned value shall be negative.

```
public short getContentCachePriority()
```

Retrieves the value of the ContentCachePriority field of the Content internal attribute.
If the content is included, an exception shall be raised.
If the field is not provided, the returned value shall be negative.
Otherwise, the returned value shall be within the range [0,255].

```
public ObjectReference mhegClone()
```

Triggers execution of the Clone elementary action.

```
public void preload()
```

Triggers execution of the Preload elementary action.

```
public void unload()
```

Triggers execution of the Unload elementary action.

C.11 Link

```
public class Link extends Ingredient
```

Methods

```
public void activate()
```

Triggers execution of the Activate elementary action.

```
public void deactivate()
```

Triggers execution of the Deactivate elementary action.

C.12 Program

```
abstract public class Program extends Ingredient
```

Methods

```
public boolean call(Object[] parameters)
```

Triggers execution of the Call elementary action.

Any element of the parameters array is expected to be of one of the following types: Boolean, Integer, OctetString, ObjectReference or ContentReference; otherwise, an exception shall be raised.

`public boolean` **`fork`**`(Object[] parameters)`

Triggers execution of the Fork elementary action.
Any element of the parameters array is expected to be of one of the following types: Boolean, Integer, OctetString, ObjectReference or ContentReference; otherwise, an exception shall be raised.
The return value specifies whether the program was correctly launched.

`public void` **`stop`**`()`

Triggers execution of the Stop elementary action.

C.13 ResidentProgram

`public class` **`ResidentProgram`** `extends Program`

C.14 RemoteProgram

`public class` **`RemoteProgram`** `extends Program`

C.15 InterchangedProgram

`public class` **`InterchangedProgram`** `extends Program`

C.16 Palette

`public class` **`Palette`** `extends Ingredient`

C.17 Font

`public class` **`Font`** `extends Ingredient`

C.18 CursorShape

`public class` **`CursorShape`** `extends Ingredient`

C.19 Variable

`abstract public class` **`Variable`** `extends Ingredient`

C.20 BooleanVariable

`public class` **`BooleanVariable`** `extends Variable`

Methods

`public void` **`setVariable`**`(boolean value)`

Sets the value of the Value internal attribute to value.
As with the attribute retrieval methods, this method accesses the value directly, as opposed to triggering the corresponding elementary action.

`public boolean` **`getVariable`**`()`

Retrieves the value of the Value internal attribute.

C.21 IntegerVariable

`public class` **`IntegerVariable`** `extends Variable`

Methods

`public void` **`setVariable`**`(int value)`
Sets the value of the Value internal attribute to value.
As with the attribute retrieval methods, this method accesses the value directly, as opposed to triggering the corresponding elementary action.

`public int` **`getVariable`**`()`
Retrieves the value of the Value internal attribute.

C.22 OctetStringVariable

`public class` **`OctetStringVariable`** `extends Variable`

Methods

`public void` **`setVariable`**`(OctetString value)`
Sets the value of the Value internal attribute to value.
As with the attribute retrieval methods, this method accesses the value directly, as opposed to triggering the corresponding elementary action.

`public OctetString` **`getVariable`**`()`
Retrieves the value of the Value internal attribute.

C.23 ObjectReferenceVariable

`public class` **`ObjectRefVariable`** `extends Variable`

Methods

`public void` **`setVariable`**`(ObjectReference value)`
Sets the value of the Value internal attribute to value.
As with the attribute retrieval methods, this method accesses the value directly, as opposed to triggering the corresponding elementary action.

`public ObjectReference` **`getVariable`**`()`
Retrieves the value of the Value internal attribute.

C.24 ContentReferenceVariable

`public class` **`ContentRefVariable`** `extends Variable`

Methods

`public void` **`setVariable`**`(ContentReference value)`
Sets the value of the Value internal attribute to value.
As with the attribute retrieval methods, this method accesses the value directly, as opposed to triggering the corresponding elementary action.

`public ContentReference` **`getVariable`**`()`
Retrieves the value of the Value internal attribute.

C.25 Presentable

`abstract public class` **`Presentable`** `extends Ingredient`

Methods

`public void` **`run`**`()`
Triggers execution of the Run elementary action.

```
public void stop()
```
Triggers execution of the Stop elementary action.

C.26 TokenGroup

```
public class TokenGroup extends Presentable
```

Methods

```
public void move(short movementId)
```
Triggers execution of the Move elementary action.

```
public void moveTo(short index)
```
Triggers execution of the MoveTo elementary action.
The index parameter value shall be within the range [0, number of elements in the group].

```
public short getTokenPosition()
```
Retrieves the value of the TokenPosition internal attribute.

```
public void callActionSlot(short index)
```
Triggers execution of the CallActionSlot elementary action.
The index parameter value shall be within the range [0, number of elements in the group].

C.27 ListGroup

```
public class ListGroup extends TokenGroup
```

Methods

```
public void addItem(short itemIndex,
                    ObjectReference visibleReference)
```
Triggers execution of the AddItem elementary action.

```
public void delItem(ObjectReference visibleReference)
```
Triggers execution of the DelItem elementary action.

```
public ObjectReference getListItem(short itemIndex)
```
Retrieves the value of the item whose rank is itemIndex (starting at 1) in the ItemList internal attribute.

```
public ObjectReference getCellItem(short cellIndex)
```
Triggers execution of the GetCellItem elementary action.

```
public boolean getItemStatus(short itemIndex)
```
Triggers execution of the GetItemStatus elementary action.

```
public void selectItem(short itemIndex)
```
Triggers execution of the SelectItem elementary action.

```
public void deselectItem(short itemIndex)
```
Triggers execution of the DeselectItem elementary action.

```
public void toggleItem(short itemIndex)
```
Triggers execution of the ToggleItem elementary action.

```
public void scrollItems(short itemsToScroll)
```
Triggers execution of the ScrollItems elementary action.

```
public void setFirstItem(short itemIndex)
```
Triggers execution of the SetFirstItem elementary action.

```
public short getFirstItem()
```
Retrieves the value of the FirstItem internal attribute.
The returned value shall be positive.

```
public short getListSize()
```
Retrieves the current number of items in the ItemList internal attribute.
The returned value shall be positive.

C.28 Visible

```
abstract public class Visible extends Presentable
```

Methods

```
public void setPosition(short xPosition,
                        short yPosition)
```
Triggers execution of the SetPosition elementary action.

```
public short[] getPosition()
```
Retrieves the value of the Position internal attribute.
The returned array shall have two elements, the first is the XPosition, the second is the YPosition.

```
public void setBoxSize(short xBoxSize,
                       short yBoxSize)
```
Triggers execution of the SetBoxSize elementary action.
The xBoxSize and yBoxSize parameter values shall be positive and different from 0.

```
public short[] getBoxSize()
```
Retrieves the value of the BoxSize internal attribute.
The returned array shall have two elements, the first is the XBoxSize, the second is the YBoxSize.

```
public void bringToFront()
```
Triggers execution of the BringToFront elementary action.

```
public void sendToBack()
```
Triggers execution of the SendToBack elementary action.

```
public void putBefore(ObjectReference visibleReference)
```
Triggers execution of the PutBefore elementary action.

```
public void putBehind(ObjectReference visibleReference)
```
Triggers execution of the PutBehind elementary action.

```
public void setPaletteRef(ObjectReference paletteReference)
```
Triggers execution of the SetPaletteRef elementary action.

```
public ObjectReference getPaletteRef()
```
Retrieves the value of the PaletteRef internal attribute.

C.29 Bitmap

```
public class Bitmap extends Visible
```

Methods

```
public void scaleBitmap(short xScale,
                        short yScale)
```
Triggers execution of the ScaleBitmap elementary action.
The xScale and yScale parameter values shall be positive.