International Telecommunication Union

ITU-T

TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU

T.172

Corrigendum 1

(06/2008)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Multimedia and hypermedia framework

MHEG-5 – Support for base-level interactive applications

Corrigendum 1

Recommendation ITU-T T.172 (1998) – Corrigendum 1

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

Facsimile – Framework	T.0–T.19
Still-image compression – Test charts	T.20–T.29
Facsimile – Group 3 protocols	T.30–T.39
Colour representation	T.40–T.49
Character coding	T.50–T.59
Facsimile – Group 4 protocols	T.60–T.69
Telematic services – Framework	T.70–T.79
Still-image compression – JPEG-1, Bi-level and JBIG	T.80–T.89
Telematic services – ISDN Terminals and protocols	T.90–T.99
Videotext – Framework	T.100–T.109
Data protocols for multimedia conferencing	T.120–T.149
Telewriting	T.150–T.159
Multimedia and hypermedia framework	**T.170–T.189**
Cooperative document handling	T.190–T.199
Telematic services – Interworking	T.300–T.399
Open document architecture	T.400–T.429
Document transfer and manipulation	T.430–T.449
Document application profile	T.500–T.509
Communication application profile	T.510–T.559
Telematic services – Equipment characteristics	T.560–T.649
Still-image compression – JPEG 2000	T.800–T.849
Still-image compression – JPEG-1 extensions	T.850–T.899

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

Recommendation ITU-T T.172

MHEG-5 – Support for base-level interactive applications

Corrigendum 1

Summary

Corrigendum 1 to Recommendation ITU-T T.172 | ISO/IEC 13522-5 provides a set of corrections, specifically the corrections of: 1) spelling errors, 2) ambiguous specifications, 3) missing specifications, and 4) erroneous specifications. The original architecture and the design policy have been maintained as defined in the original specification.

Source

Corrigendum 1 to Recommendation ITU-T T.172 (1998) was approved on 13 June 2008 by ITU-T Study Group 16 (2005-2008) under Recommendation ITU-T A.8 procedure.

CONTENTS

Page

Introduction

This corrigendum is a response to public comments to ISO/IEC JTC 1/SC 29/WG 12 from ongoing implementation work with the MHEG-5 standard. The comments contribute to identify several issues of MHEG-5 usage:

1) Spelling errors
2) Ambiguous specification
3) Missing specification
4) Erroneous specification

These issues have been collected by JTC 1/SC 29/WG 12, and the JTC 1/SC 29/WG 12 MHEG-5 Maintenance Task Force (MTF) which was established to deal with the input properly. The work of the MTF concentrates to keep the spirit of MHEG-5 specification and only to "repair" problems in the published MHEG-5 text after deep technical evaluation. Requests for additional functionality have been rejected as long as they do not help to remove problems in the text. Instead, additional specification was added only when missing, and clarification of the intention of the original MHEG-5 text was added where required. In case of ambiguous text, a conservative technical solution was preferred.

This corrigendum is organized according to the MHEG-5 class hierarchy. The subclauses for each class describe the separate issues, and identify the requested action which solve the issue.

Recommendation ITU-T T.172

MHEG-5 – Support for base-level interactive applications

Corrigendum 1

1 Notations

The following notations are used in the following subclauses to describe the changes defined in this corrigendum to Rec. ITU-T T.172 | ISO/IEC 13522-5.

<Name of the class>

1.1 Name of the issue

1.1.1 Description

This subclause describes the problem in Rec. ITU-T T.172 | ISO/IEC 13522-5 which is addressed in this issue.

1.1.2 Required action

This subclause describes the kind of changes to Rec. ITU-T T.172 | ISO/IEC 13522-5 which solve the problem described in the previous subclause.

2 Root Class

2.1 Misleading note in Activation behaviour

2.1.1 Description

The first Note of the *Activation* behaviour of the Root class (subclause 8.3) is misleading.

2.1.2 Required action

Remove the first Note of the *Activation* behaviour of the Root class (subclause 8.3).

2.2 Generation of the ContentAvailable event

2.2.1 Description

A clarification for the generation of the *ContentAvailable* event is needed for objects which have no encoded Content attribute.

2.2.2 Required action

Add the following Note to the description of the *ContentAvailable* event (subclause 8.2) of the Root class:

"

NOTE – Objects which have no encoded Content attribute generate no *ContentAvailable* event.

"

2.3 Conditions to set the AvailabilityStatus to False

2.3.1 Description

The description of the preparation behaviour is not symmetrical to the destruction behaviour (see subclause 8.3, *Destruction* behaviour).

2.3.2 Required action

Add the following step to the sequence of actions after step 5 in the Root *Destruction* behaviour, and renumber the original step 6 to step 7:

"

6) Set the *AvailabilityStatus* to False.

"

2.4 Introduction of the ContentPreparation behaviour in the ContentAvailable event

2.4.1 Description

A new behaviour *ContentPreparation* is introduced in the Root class (subclause 8.3) to clarify the process of content retrieval.

2.4.2 Required action

Change the first paragraph of the description of the *ContentAvailable* event in subclause 8.2 to:

"

This event is generated when the object and its content are available in an optimal state to the engine. This event is generated asynchronously with the *ContentPreparation* behaviour for the object.

"

2.5 Definition of the ContentPreparation behaviour

2.5.1 Description

The new behaviour *ContentPreparation* is introduced in the Root class (subclause 8.3) to clarify the process of content retrieval.

2.5.2 Required action

Add the following behaviour to the internal behaviours of the Root class in subclause 8.3:

"

ContentPreparation	This behaviour has the basic semantics of loading and processing all requested resources in order to handle or to present this object. This behaviour performs no action in the Root class.

"

2.6 Clarification of the Preparation behaviour

2.6.1 Description

The new behaviour *ContentPreparation* is introduced in the Root class (subclause 8.3) to clarify the process of content retrieval. For this, the *Preparation* behaviour of the Root class must be adapted.

2.6.2 Required action

Remove the instruction between step 5 and 6 of the *Preparation* behaviour of the Root class in subclause 8.3 and change point 6 to:

"

6) Perform the *ContentPreparation* behaviour.

"

3 Group Class

3.1 Ambiguous situations during context changes

3.1.1 Description

The use of elementary actions which change the context (i.e., Launch, Spawn, TransitionTo, Quit) might lead to ambiguous situations if they are used in the OnStartup, OnSpawnCloseDown, OnCloseDown, and OnRestart attributes (see also subclause 4.3 of this corrigendum).

3.1.2 Required action

Add the following Note to the definition of the OnStartup and OnCloseDown attributes of the Group class (subclause 9.1.2):

"

NOTE – The elementary actions Launch, Spawn, TransitionTo, and Quit shall be ignored during the execution of this attribute.

"

4 Application Class

4.1 Clarification of the description of Spawn in case of limited ApplicationStack

4.1.1 Description

The last sentence of the Spawn elementary action description is not clear.

4.1.2 Required action

Replace the last sentence of the Spawn elementary action description (subclause 10.4) in the Application class with the following text:

"

If it is not implemented, or if the application identifier stack is full, ignore the steps described above and execute the Launch elementary action instead.

"

4.2 Targeting Launch and Spawn elementary actions to the currently active Application

4.2.1 Description

The description of the Spawn elementary action is not sufficient for the case that the target of the elementary action is the currently active Application.

4.2.2 Required action

Add the following Note to the description of the Spawn elementary action (subclause 10.4) in the Application class:

"

NOTE – If the target Application is available, ignore the action.

"

4.3 Ambiguous situations during context changes

4.3.1 Description

The use of elementary actions which change the context (i.e., Launch, Spawn, Quit, TransitionTo) might lead to ambiguous situations if they are used in the OnStartup, OnSpawnCloseDown, OnCloseDown, and OnRestart attributes (see also subclause 3.1 of this corrigendum).

4.3.2 Required action

Add the following Note to the definitions of the OnSpawnCloseDown and OnRestart attributes of the Application class (subclause 10.1.2):

"

> NOTE – The elementary actions Launch, Spawn, TransitionTo, and Quit shall be ignored during the execution of this attribute.

"

4.4 Handling of events during context changes

4.4.1 Description

The Note in subclause 10.4 for the Launch elementary action is a contradiction to the text in subclause 53.3.

4.4.2 Required action

Remove the Note for the Launch elementary action in subclause 10.4.

4.5 Missing specification of the behaviour in case of a failed Quit elementary action

4.5.1 Description

The description of the Quit elementary action does not specify the behaviour in case of a failing Quit elementary action. This is a special case since this elementary action results in retrieval of an object reference from the application stack, which may not be a valid object reference.

4.5.2 Required action

Add the following Note to the description of the Quit elementary action (subclause 10.4) in the Application class:

"

> NOTE – If the object reference retrieved from the application stack is not a valid object reference and the Quit elementary action fails due to that, it is up to the application domain to define the handling of this situation.

"

4.6 Wrong order of execution of the actions in the OnRestart attribute

4.6.1 Description

The elementary actions in the OnRestart attribute (subclause 10.1.2) are executed after the elementary actions of the OnStartUp attribute, after all initially active objects in the Application object have been activated, and after a possible transition to a Scene object (including the activation of the initially active objects of the Scene) has happened.

4.6.2 Required action

Remove step 6 of the description of the Quit elementary action (subclause 10.4) and add to step 4 of its description:

"

> Execute the OnRestart elementary action of the newly activated Application object between step 4 and 5 of the *Activation* behaviour of the Group class.

"

5 Scene Class

5.1 Clarification for SetTimer with AbsoluteTime

5.1.1 Description

The description of the SetTimer elementary action does not specify how to handle an absolute time which has already passed.

5.1.2 Required action

Add the following text to the description of the SetTimer elementary action (subclause 11.4) in the Scene class:

"

> If the time indicated in *TimerValue* has already passed and the *AbsoluteTime* is set to True, the *TimerFired* event shall not be raised.

"

5.2 Missing EmulatedEventType

5.2.1 Description

In the formal syntax description of the SendEvent elementary action (subclause 11.4), the *EngineEvent* type is missing in the list of *EmulatedEventTypes*, but it is allowed in the ASN.1 and textual notation syntax definitions.

5.2.2 Required action

Add the *EngineEvent* to the list of *EmulatedEventTypes* to the formal syntax definition of the SendEvent elementary action in subclause 11.4.

5.3 Clarification on the Provisions of Use of the SendEvent elementary

5.3.1 Description

A clarification for the definition of the scope of the *EmulatedEventSource* parameter of the SendEvent elementary action in subclause 11.4 is needed.

5.3.2 Required action

Replace the second bullet of the Provisions of Use of the SendEvent elementary action with the following text:

"

- *EmulatedEventSource* shall refer to an object in the current scope. This object needs to be an instance of a class that is capable of generating an event of type *EmulatedEventType*.

"

6 Ingredient Class

6.1 Mandatory Content Hook

6.1.1 Description

The ContentHook exchanged attribute of the Ingredient Class is not mandatory, neither in the Ingredient class itself nor in the DefaultAttributes exchanged attribute of the Application class. The second point of the Provisions of Use of the SetData elementary action of the Ingredient class does not specify where the ContentHook shall be defined.

6.1.2 Required action

The second point of the Provisions of Use of the SetData elementary action (subclause 12.4) in the Ingredient class shall read as follows:

"

- The ContentHook of the target Ingredient object shall be specified by means of the corresponding Application default or the encoded hook value.

"

6.2 Wrong initial value for the ContentReference field of the Content attribute

6.2.1 Description

The definition of the initial value of the ContentReference field of the Content attribute (subclause 12.1.3) is wrong.

6.2.2 Required action

Change the definition of the initial value of the ContentReference field of the Content attribute (subclause 12.1.3) to:

"

- Initial value: ContentReference of OriginalContent attribute.

"

6.3 SetData elementary action targeted to "InitiallyActive False" objects

6.3.1 Description

SetData can only be targeted to available objects. The solution is to target the Preload elementary action to the target object to execute its preparation behaviour.

6.3.2 Required action

Add the following Note to the SetData elementary action (subclause 12.4) in the Ingredient class:

"

NOTE – The Preload elementary action can be used to make a non-available Ingredient object available.

"

6.4 Targeting Preload and Unload elementary actions to object without content data

6.4.1 Description

The Preload and Unload elementary actions can only be targeted to Ingredient objects which have an encoded Content attribute. At least, the Preload elementary action is used to prepare objects in advance and possibly to put them into the display stack. This is not possible for Visible objects which have no Content attribute, e.g., Rectangle objects.

6.4.2 Required action

Remove the second part of the Provisions of Use of both the descriptions of the Preload and Unload elementary actions in subclause 12.4.

Change the last sentence of the Clone elementary action (subclause 12.4) to:

"

A dynamically created Ingredient can also be destroyed using the Unload elementary action.

"

6.5 Clarification for the presentation status of Presentable objects

6.5.1 Description

A clarification about the presentation status of initially active Presentable objects is missing in the description of the Ingredient class (clause 12).

6.5.2 Required action

Add the following explicit *Activation* behaviour to the Ingredient class (subclause 12.3):

"

Activation: Apply the *Activation* behaviour as inherited from the base class.

"

6.6 Introduction of the ContentPreparation behaviour

6.6.1 Description

The new behaviour *ContentPreparation* introduced in the Root class is also introduced for the Ingredient class (subclause 12.3) to clarify the process of content retrieval.

6.6.2 Required action

Add the following behaviour to the internal behaviours of the Ingredient class in subclause 12.3:

"

ContentPreparation Apply the following sequence of actions *synchronously*:

1) If the Ingredient does not have the *Content* attribute encoded, then ignore this action.
2) Cancel any outstanding *ContentPreparation* asynchronous steps for this object.
3) Initiate retrieval of the data for the *Content* attribute of the object.

The following step is *asynchronous* and occurs when the content of the object has been fully retrieved:

4) Generate a *ContentAvailable* event.

"

6.7 Cancellation of outstanding content retrieval requests in the Destruction behaviour

6.7.1 Description

When an Ingredient object is destructed, any outstanding content retrieval requests should be cancelled.

6.7.2 Required action

Insert a new step before all other steps in the *Destruction* behaviour of subclause 12.3 of the Ingredient class:

"

1) Cancel any outstanding *ContentPreparation* asynchronous steps for this object.

"

Renumber the following steps accordingly.

6.8 Modification of the SetData elementary action

6.8.1 Description

In order to use the *ContentPreparation* behaviour of the Ingredient class defined in subclause 12.3, the SetData elementary action in subclause 12.4 needs to be modified.

6.8.2 Required action

Change the description of the SetData elementary action of the Ingredient class in subclause 12.4 to:

"

Execute the following sequence of actions:

1) Set the *Content* attribute of the target Ingredient to *NewContent*.
2) Apply the *ContentPreparation* behaviour.

"

Leave the Provisions of Use unchanged.

6.9 Clarification of the Preload elementary action

6.9.1 Description

A clarification is needed for the Preload elementary action in subclause 12.4 for the optional loading of content.

6.9.2 Required action

Change the sequence of instructions in the second paragraph of the Preload elementary action in subclause 12.4 to:

"

Apply the *Preparation* behaviour.

"

7 Program Class

7.1 Program calls and return values

7.1.1 Description

It is not clear from Rec. ITU-T T.172 | ISO/IEC 13522-5 how return values of a program call are to be returned to the calling application. Values are to be returned in variables. There are two methods possible how to pass the variables to the called program. One is call-by-value of the content (i.e., an object reference to the variable is passed), the other is call-by-reference (i.e., the variable is directly passed to the program).

7.1.2 Required action

Add the following bullet to the end of the Provisions of Use of the Call and Fork elementary actions (subclause 14.4) of the Program class:

"

- Output parameters and Input/Output parameters shall be encoded as follows: The IndirectReference option in GenericBoolean, GenericInteger, GenericOctetString, GenericObjectReference, and GenericContentReference respectively shall be encoded.

"

7.2 Clarification for passing input parameters to Programs

7.2.1 Description

A clarification is needed for passing input parameters to Programs in regards to the asynchronous execution of Programs.

7.2.2 Required action

Add the following sentence to the 3rd point of the *Activation* behaviour in subclause 14.3 of the Program class:

"

All input parameters shall be passed by value.

"

7.3 Changing Variable objects during the asynchronous execution of a Program

7.3.1 Description

The Note in the definition before the Provisions of Use in the description of the Fork elementary action (subclause 14.4) defines that an MHEG-5 application must neither access the Variable object referenced by *ForkSucceeded* nor those Variables of *Parameters* which are output parameters before the *AsynchStopped* event has been dealt with.

7.3.2 Required action

Change the sentence after the first sequence of instructions of the Fork elementary action in subclause 14.4 to:

"

> When the execution of the external procedural code finishes, execute the following sequence of actions without interruption:

"

Add a second Note to the definition of the Fork elementary action (subclause 14.4):

"

> NOTE 2 – See subclause 53.3 regarding timing of updating variables.

"

8 RemoteProgram Class

8.1 Double execution of parts of the RemoteProgram Class Activation behaviour

8.1.1 Description

The *Activation* behaviour in subclause 16.3 of the RemoteProgram class shall not call for duplication of actions.

8.1.2 Required action

Replace step 3 of the *Activation* behaviour in 16.3 by the following:

"

> 3) Apply steps 3 to 7 from the *Activation* behaviour as inherited from the base class.

"

9 Palette Class

9.1 Missing activation behaviour for the Palette class

9.1.1 Description

Objects of the Palette class are never activated. There is no *Activation* behaviour which sets the *RunningStatus* internal attribute to True.

9.1.2 Required action

Add the following *Activation* behaviour to the internal behaviours of the Palette (subclause 18.3) class:

"

Activation	1) Apply the *Activation* behaviour as defined in the base class.
	2) Set the *RunningStatus* attribute to True and generate an *IsRunning* event.

"

10 Font Class

10.1 Missing activation behaviour for the Font class

10.1.1 Description

Objects of the Font class are never activated. There is no *Activation* behaviour which sets the *RunningStatus* internal attribute to True.

10.1.2 Required action

Add the following *Activation* behaviour to the internal behaviours of the Font (subclause 19.3) class:

"

Activation	1)	Apply the *Activation* behaviour as defined in the base class.
	2)	Set the *RunningStatus* attribute to True and generate an *IsRunning* event.

"

11 CursorShape class

11.1 Missing activation behaviour for the CursorShape class

11.1.1 Description

Objects of the CursorShape class are never activated. There is no *Activation* behaviour which sets the *RunningStatus* internal attribute to True.

11.1.2 Required action

Add the following *Activation* behaviour to the internal behaviours of the CursorShape (subclause 20.3) class:

"

Activation	1)	Apply the *Activation* behaviour as defined in the base class.
	2)	Set the *RunningStatus* attribute to True and generate an *IsRunning* event.

"

12 Presentable Class

12.1 Removal of the SetData elementary action

12.1.1 Description

Since a new internal behaviour *ContentPreparation* is introduced for the Ingredient class in subclause 12.3, the modified definition of the SetData elementary action can be removed.

12.1.2 Required action

Remove the definition of the SetData elementary action from subclause 27.4.

12.2 Modification of the introductory text for subclause 27.4

12.2.1 Description

Since the new definition of the SetData elementary action is removed from the Presentable class, the introduction to subclause 27.4 needs to be modified.

12.2.2 Required action

Change the introduction to subclause 27.4 to:

"

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

"

13 TokenManager Class

13.1 Misleading example for the MovementTable

13.1.1 Description

The example given for the Movements in the description of the *MovementTable* exchanged attribute of the TokenManager Class (subclause 28.1.2) is misleading.

13.1.2 Required action

Remove the Movement Table Example after the bullet list in subclause 28.1.2.

A better example is presented in Appendix III (see clause 32 of this corrigendum).

13.2 Missing Note in the description of the TokenManager class

13.2.1 Description

Appendix III describes the usage of the TokenManager class in greater detail. A Note which points to this appendix is missing in the description of the class.

13.2.2 Required action

Add the following Note before subclause 28.1:

"

NOTE – Refer to Appendix III for further details.

"

13.3 Misleading expression in the description of the MovementTable

13.3.1 Description

The example in the fourth sentence of the description of the MovementTable uses the word "number" instead of "index" which is misleading.

13.3.2 Required action

Change the fourth sentence of the description of the MovementTable in subclause 28.1.2 to:

"

For instance, if the token is on element 2, and the elementary action Move(4) is executed, the expression *f*(2, 4) evaluates to the index of the element to get the token.

"

13.4 Clarification for the range of the MovementIdentifier parameter of the Move elementary action

13.4.1 Description

A further clarification is needed for the Move elementary action in subclause 28.4.

13.4.2 Required action

Add the following bullet to the Provisions of Use of the Move elementary action in subclause 28.4:

"

- The *MovementTable* exchanged attribute shall be encoded for this TokenManager, and the *MovementIdentifier* shall be in the range [1,M], where M is the number of *Movement* structures encoded in the *MovementTable* exchanged attribute.

"

14 ListGroup Class

14.1 Clarification in the description of the ListGroup class

14.1.1 Description

The introductory description of the ListGroup class is misleading and needs clarification.

14.1.2 Required action

Change the first sentence of the introductory description of the ListGroup class in clause 30 to:

"

This class defines locations on the screen (cells) for the set of elements managed by TokenManager.

"

Add the following Notes to the Notes of the introduction (before subclause 30.1):

"

NOTE 3 – The author should be aware of possible unexpected results if the ListGroup has unused cells. It is his responsibility to prevent the token being held by an unused cell, if this is not desirable.

NOTE 4 – Refer to Appendix III for further details.

"

14.2 Clarification for inherited attributes

14.2.1 Description

Some of the inherited attributes of the ListGroup class need clarifications.

14.2.2 Required action

Change subclause 30.1.1 (Inherited attributes) to the following text:

"

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

Attribute name	Defined in	Constraints and requirements
TokenGroupItems	TokenGroup	This attribute contains a set of Visibles to be used as the initial contents of the *ItemList* and the ActionSlots associated with the Cells of the ListGroup. The AVisible attribute of each TokenGroupItem may contain the Null ObjectReference.
TokenPosition	TokenManager	In the ListGroup class, this attribute shall take values only in the range *[*0,*N]*, where *N* is the number of Cells in the ListGroup. The value 0 signifies that no Cell has the token.

NOTE – There is no connection between the Visibles defined in the TokenGroupItems and the ActionSlots. The first N sets of ActionSlots (where N is the number of Cells) are associated with the Cells defined by the *Positions* attributes. The AVisible attributes are only used to set the initial contents of the *ItemList*.

"

14.3 Misleading description of the Positions exchanged attribute

14.3.1 Description

The description of the *Positions* exchanged attribute in subclause 30.1.2 is misleading.

14.3.2 Required action

Change the description of the *Positions* exchanged attribute in subclause 30.1.2 to:

"

Positions	Set of screen coordinates defining the Cells of the ListGroup. Sequence of the following data structure: • Position: pair of integers (XPosition, YPosition).

"

14.4 Wrong description of the Deselect behaviour

14.4.1 Description

The description parts 2 and 3 of the *Deselect* behaviour of the ListGroup class (subclause 30.3) are wrong.

14.4.2 Required action

Change the description of parts 2 and 3 of the *Deselect* behaviour of the ListGroup class to:

"

2) Set the *ItemSelectionStatus* of the item with the index *ItemIndex* to False.

3) Generate an *ItemDeselected* event with *ItemIndex* as associated data.

"

14.5 Spelling error in the description of the ItemList internal attribute

14.5.1 Description

There is a spelling error in the second paragraph of the description of the *ItemList* internal attribute of the ListGroup class (subclause 30.1.3)

14.5.2 Required action

Change the first sentence of the second paragraph of the *ItemList* internal attribute of the ListGroup class in subclause 30.1.3 to:

"

Each item in the *ItemList* has an *ItemSelectionStatus* attribute.

"

14.6 Clarification of the FirstItem internal attribute

14.6.1 Description

A clarification is needed for the relationship between the *FirstItem* and *WrapAround* internal attributes of the ListGroup class (subclause 30.1.3). The examples given in the description are misleading.

14.6.2 Required action

Change the definition of the *FirstItem* internal attribute (subclause 30.1.3) to:

"

FirstItem	The index of the item of the *ItemList* which is presented at the first cell. This defines a 'window' on the ordered list of items in the *ItemList*. This window is equal in size to the number of cells and the position of this window with respect to the items that can be changed using the *ScrollItems* elementary action. The presentation of the items in the list depends on the position of this window, and the value of the *WrapAround* attribute. Refer to Appendix III for a detailed explanation of this behaviour. FirstItem may take any Integer value in the range [1, length of *ItemList*] If the *ItemList* is empty, then *FirstItem* shall take the value 1. • Integer. • Initial value: 1.

"

14.7 Clarification of the FirstItemPresented event

14.7.1 Description

A clarification is needed for the FirstItemPresented event and the change of the presentation status of the first item of the *ItemList* internal attribute.

14.7.2 Required action

Replace the description of the FirstItemPresented event (subclause 30.2) with the following text:

"

FirstItemPresented	This event is generated with the associated data set to True when the first item in the *ItemList* becomes presented at a cell in the ListGroup. This event is generated with the associated data set to False when the first item in the *ItemList* ceases to be presented at a cell in the ListGroup, or when the list becomes empty. The presentation status of the item can change if the *FirstItem* attribute is changed, or if the number of items in the list is changed. Associated data: Boolean. True if the item is presented, False if it is not presented. NOTE – This event is intended to indicate whether the head of the *ItemList* occupies a cell. Therefore, the event is not generated when the first item in the *ItemList* changes, but the visibility of the head of the list does not change.

"

14.8 Clarification of the LastItemPresented event

14.8.1 Description

A clarification is needed for the LastItemPresented event and the change of the presentation status of the last item of the *ItemList* internal attribute.

14.8.2 Required action

Replace the description of the LastItemPresented event (subclause 30.2) with the following text:

"

LastItemPresented	This event is generated with the associated data set to True when the last item in the *ItemList* becomes presented at a cell in the ListGroup. This event is generated with the associated data set to False when the last item in the *ItemList* ceases to be presented at a cell in the ListGroup, or when the list becomes empty. The presentation status of the item can change if the *FirstItem* attribute is changed, or if the number of items in the list is changed. Associated data: Boolean. True if the item is presented, False if it is not presented. NOTE – This event is intended to indicate whether the base of the *ItemList* occupies a cell. Therefore, the event is not generated when the last item in the *ItemList* changes, but the visibility of the base of the list does not change.

"

14.9 Clarification for the HeadItems event

14.9.1 Description

The description of the HeadItems event in subclause 30.2 is not very clear.

14.9.2 Required action

Change the first sentence of the description of the HeadItems event in subclause 30.2 to:

"

This event is generated each time the number of items in the *ItemList* with an index smaller than *FirstItem* changes.

"

14.10 Clarification for the TailItems event

14.10.1 Description

The description of the TailItems event in subclause 30.2 is not very clear.

14.10.2 Required action

Change the first sentence of the description of the TailItems event in subclause 30.2 to:

"

This event is generated each time the number of items in the *ItemList* with an index greater than or equal to *FirstItem* changes.

"

Add the following sentence to the end of the description of the TailItems event before the bullets in subclause 30.2:

"

When the list becomes empty, the event is generated with value 0.

"

14.11 Clarification for the Preparation behaviour

14.11.1 Description

The definition of the *Preparation* behaviour does not define the behaviour in case of multiple specified visible objects.

In addition, a description is needed for the *Preparation* behaviour that Null ObjectReferences shall not be added to the *ItemList* internal attribute of the ListGroup class (subclause 30.3).

14.11.2 Required action

Change point 2 of the definition of the *Preparation* behaviour of the ListGroup class (subclause 30.3) to:

"

2) Add each reference listed in the *TokenGroupItems* attribute to the *ItemList* in the order they are listed in the *TokenGroupItems* attribute. If a Visible is referenced more than once in the *TokenGroupItems*, it is added only at its first occurrence in the *ItemList*. Null ObjectReferences are not added to *ItemList*.

"

14.12 Clarification for non-available Visibles in the Destruction behaviour

14.12.1 Description

A clarification is needed for the *Destruction* behaviour of the ListGroup and the handling of non-available Visible objects.

14.12.2 Required action

Change the first point of the definition of the *Destruction* behaviour of the ListGroup class (subclause 30.3) to:

"

1) Reset all available Visibles of the ListGroup to their OriginalPosition.

"

14.13 Clarification for the Activation behaviour

14.13.1 Description

A clarification is needed for the *Activation* behaviour of the ListGroup and the generation of the *TokenMovedTo* event.

14.13.2 Required action

Change the *Activation* behaviour of the ListGroup class (subclause 30.3) to:

"

Activation Execute the following sequence of actions:

1) Apply the *Activation* behaviour as inherited from the Presentable class.
2) Generate a *TokenMovedTo* event with the value of the *TokenPosition* attribute as associated data.
3) Set the *RunningStatus* to True.
4) Apply the *Update* behaviour.
5) Generate an *IsRunning* event.

"

14.14 Clarification for non-active Visibles in the Deactivation behaviour

14.14.1 Description

A clarification is needed for the *Deactivation* behaviour of the ListGroup and the handling of non-active Visible objects.

14.14.2 Required action

Change the first point of the definition of the *Deactivation* behaviour of the ListGroup class (subclause 30.3) to:

"

1) Apply the *Deactivation* behaviour to all active Visibles referenced in the *ItemLis*t.

"

14.15 Clarification of the Update behaviour

14.15.1 Description

The *Update* behaviour needs further clarifications regarding the changes presented in subclauses 14.9, 14.12, and 14.13 of this corrigendum.

14.15.2 Required action

Change the *Update* behaviour of ListGroup class (subclause 30.3) to the following text:

"

Update	If the *RunningStatus* of the ListGroup is True, then execute the following sequence of actions:

1) For each item that will not be presented in a cell, if the item is active, then apply the *Deactivation* behaviour to it and set its position to its *OriginalPosition* attribute.
2) For each item, if it is to be presented at a cell, apply the *Preparation* behaviour to it and then set its *Position* (internal attribute) to the position defined for that cell.
3) For each item that will be presented in a cell, if the item is inactive, apply the *Activation* behaviour to it.

"

14.16 New internal behaviour for adjusting an index

14.16.1 Description

For various elementary actions, an adjustment of an index parameter is needed in order to point to the correct position.

14.16.2 Required action

Add the following internal behaviour to subclause 30.3:

"

AdjustIndex (Index, Adjusted)	Set the *Adjusted* parameter to the value of the *Index* parameter normalized to the length of the *ItemList*, by applying the following logic:

If the length of *ItemList* is zero then set *Adjusted* to 1.

Otherwise, if *Index* is greater than the length of *ItemList*, set *Adjusted* to:

((*Index*–1) MODULO Length of *ItemList*) + 1

Otherwise, if *Index* is less than 1, set *Adjusted* to:

Length of *ItemList* – ((–*Index*) MODULO Length of *ItemList*)

Otherwise, set *Adjusted* to *Index*, unmodified.

"

14.17 Change of the AddItem internal behaviour

14.17.1 Description

A clarification is needed for the *AddItem* internal behaviour in subclause 30.3.

14.17.2 Required action

Change the second point of the description of the *AddItem* internal behaviour in subclause 30.3 to:

"

2) If *Index* is less than 1 or greater than the current length of *ItemList* + 1, ignore this behaviour.

"

Add a new point 4 after point 3:

"

4) If *Index* is less than or equal to *FirstItem*, and *FirstItem* is strictly less than the new length of the *ItemList*, then increment *FirstItem* by one.

"

The old point 4 is now point 5.

14.18 Change of the Delitem internal behaviour

14.18.1 Description

A clarification is needed for the *Delitem* internal behaviour in subclause 30.3.

14.18.2 Required action

Add a new point 4 after point 3 of the *Delitem* internal behaviour in subclause 30.3:

"

4) If the index of the deleted *Item* was less than or equal to *FirstItem* and *FirstItem* is strictly greater than one, then decrease *FirstItem* internal attribute by one.

"

The old point 4 is now point 5.

14.19 Clarification for the ItemIndex parameter of the GetListItem elementary action

14.19.1 Description

A clarification is needed for the relationship between the *ItemIndex* parameter of the GetListItem elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.19.2 Required action

Change the description and the Provisions of Use of the GetListItem elementary action in subclause 30.4 to:

"

Using a local variable *Index*:

If *WrapAround* is False, if *ItemIndex* is in the range [1, length of *ItemList*], then set *Index* to *ItemIndex*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*ItemIndex*, *Index*) behaviour.

Return the reference included in the *ItemList* attribute with the *index* specified by the Index parameter in the ObjectRefVariable referenced by *ItemRefVar*.

Provisions of use:

- The *Target* object shall be an available ListGroup object.
- *ItemRefVar* shall refer to an active ObjectRefVariable object.

"

14.20 Clarification for the ItemIndex parameter of the GetItemStatus elementary action

14.20.1 Description

A clarification is needed for the relationship between the *ItemIndex* parameter of the GetItemStatus elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.20.2 Required action

Change the description and the Provisions of Use of the GetItemStatus elementary action in subclause 30.4 to:

"

Using a local variable Index:

If *WrapAround* is False, if *ItemIndex* is in the range [1, length of *ItemList*], then set *Index* to *ItemIndex*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*ItemIndex*, *Index*) behaviour.

Return the value of the *ItemSelectionStatus* attribute of the item in the *ItemList* with index *Index* in the BooleanVariable referenced by *ItemStatusVar*.

Provisions of use:

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

"

14.21 Clarification for the ItemIndex parameter of the SelectItem elementary action

14.21.1 Description

A clarification is needed for the relationship between the *ItemIndex* parameter of the SelectItem elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.21.2 Required action

Change the description of the SelectItem elementary action in subclause 30.4 to:

"

Using a local variable Index:

If *WrapAround* is False, if *ItemIndex* is in the range [1, length of *ItemList*], then set *Index* to *ItemIndex*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*ItemIndex*, *Index*) behaviour.

Apply the Select(*Index*) internal behaviour.

"

14.22 Clarification for the ItemIndex parameter of the DeselectItem elementary action

14.22.1 Description

A clarification is needed for the relationship between the *ItemIndex* parameter of the DeselectItem elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.22.2 Required action

Change the description of the DeselectItem elementary action in subclause 30.4 to:

"

Using a local variable Index:

If *WrapAround* is False, if *ItemIndex* is in the range [1, length of *ItemList*], then set *Index* to *ItemIndex*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*ItemIndex*, *Index*) behaviour.

Apply the Deselect(*Index*) internal behaviour.

"

14.23 Clarification for the ItemIndex parameter of the ToggleItem elementary action

14.23.1 Description

A clarification is needed for the relationship between the *ItemIndex* parameter of the ToggleItem elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.23.2 Required action

Change the description of the ToggleItem elementary action in subclause 30.4 to:

"

Using a local variable Index:

If *WrapAround* is False, if *ItemIndex* is in the range [1, length of *ItemList*], then set *Index* to *ItemIndex*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*ItemIndex*, *Index*) behaviour.

If the *ItemSelectionStatus* of the item indicated by *Index* is True, apply the Deselect(*Index*) internal behaviour; otherwise, apply the Select(*Index*) internal behaviour.

"

14.24 Clarification for the ItemsToScroll parameter of the ScrollItems elementary action

14.24.1 Description

A clarification is needed for the relationship between the *ItemsToScroll* parameter of the ScrollItems elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.24.2 Required action

Change the description of the ScrollItems elementary action in subclause 30.4 to:

"

If *WrapAround* is False, if *FirstItem* + *ItemsToScroll* is in the range [1, length of *ItemList*], then set *FirstItem* to *FirstItem* + *ItemsToScroll*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*FirstItem* + *ItemsToScroll*, *FirstItem*) behaviour.

Apply the *Update* behaviour.

"

14.25 Clarification for the NewFirstItem parameter of the SetFirstItem elementary action

14.25.1 Description

A clarification is needed for the relationship between the *NewFirstItem* parameter of the SetFirstItem elementary action and the *WrapAround* internal attribute of the ListGroup class (subclause 30.4).

14.25.2 Required action

Change the description of the SetFirstItem elementary action in subclause 30.4 to:

"

If *WrapAround* is False, if *NewFirstItem* is in the range [1, length of *ItemList*], then set *FirstItem* to *NewFirstItem*; otherwise, ignore this action.

If *WrapAround* is True, apply the AdjustIndex(*NewFirstItem*, *FirstItem*) behaviour.

Apply the *Update* behaviour.

"

15 Visible Class

15.1 Ambiguous description of the OriginalBoxSize attribute and the SetBoxSize elementary action

15.1.1 Description

The description of the OriginalBoxSize attribute of the Visible class (subclause 31.1.2) and the Provisions of Use of the SetBoxSize elementary action (subclause 31.4) are ambiguous in the case of zero values for the sizes of the bounding box.

15.1.2 Required action

Remove first sentence of the second paragraph of the OriginalBoxSize attribute (subclause 31.1.2). Add another point to the description of the OriginalBoxSize attribute:

"

- XBoxSize and YBoxSize shall be zero or greater.

"

Change the second bullet of the SetBoxSize elementary action (subclause 31.4) to:

"

- *XNewBoxSize* and *YNewBoxSize* shall be zero or greater.

"

15.2 Introduction of the ContentPreparation behaviour

15.2.1 Description

A new behaviour *ContentPreparation* introduced in the Root and Ingredient classes (subclause 12.3) is also introduced for the Visible class (subclause 31.3) to clarify the process of content retrieval.

15.2.2 Required action

Add the following behaviour to the internal behaviours of the Visible class in subclause 31.3:

"

ContentPreparation	Apply steps 1 to 3 of the *ContentPreparation* behaviour of the base class *synchronously*. The following steps are *asynchronous* and occur when the content of the object is in an optimal state to be displayed: 4) If the object is being displayed, then display the object again with the value of *Content* immediately. 5) Generate a *ContentAvailable* event.

"

15.3 Clarification for the Activation behaviour

15.3.1 Description

Since a new behaviour *ContentPreparation* has been introduced, the *Activation* behaviour of the Visible class needs to be modified.

15.3.2 Required action

Change the second point of the *Activation* behaviour of the Visible class in subclause 31.3 to:

"

2) Start displaying the Visible according to its position in the *DisplayStack* and to the position and the bounding box defined by the *Position* and *BoxSize* attributes.

"

Add the following Note to the end of the description:

"

NOTE – This part of Rec. ITU-T T.172 | ISO/IEC 13522-5 does not define the appearance of the Visible if it is displayed before the *Content* of the Visible is fully available.

"

15.4 Wrong definition in the GetBoxSize elementary action

15.4.1 Description

The definition of the GetBoxSize elementary action contains a spelling error.

15.4.2 Required action

Change the second paragraph of the GetBoxSize elementary action in subclause 31.4 to:

"

Set the Variables referenced by *XBoxSizeVar* and *YBoxSizeVar* to the value of the X and Y box size of the target Visible respectively.

"

16 Bitmap Class

16.1 Missing definition for the SetData elementary action on scaled Bitmap objects

16.1.1 Description

There is no indication of what the scale of a bitmap is when initially loaded. In the case of a SetData elementary action, it is not clear whether the scaling factor be retained or the bitmap is reset to the original size.

16.1.2 Required action

Add the following definition to subclause 32.4:

SetData	NOTE – A SetData elementary action resets the scaling factor of the bitmap to its original value.

17 LineArt Class

17.1 Wrong introductory description of the LineArt internal behaviours

17.1.1 Description

The introductory description of the internal behaviour of the LineArt class (subclause 33.3) is wrong.

17.1.2 Required action

Change the description of the internal behaviour of the LineArt class in subclause 33.3 to:

"

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

"

17.2 Wrong description of the OriginalRefFillColour exchanged and RefFillColour internal attributes

17.2.1 Description

Both the descriptions of *OriginalRefFillColour* exchanged and *RefFillColour* internal attributes of the LineArt class are ambiguous.

17.2.2 Required action

Change the first sentence of the definition of the *OriginalRefFillColour* exchanged attribute in subclause 33.1.2 to:

"

Initial reference colour for the background and inside of the LineArt object.

"

Change the first sentence of the definition of the *RefFillColour* internal attribute in subclause 33.1.3 to:

"

Reference colour for the background and inside of the LineArt object.

"

18 DynamicLineArt Class

18.1 Unspecified ranges for the EllipseWidth and EllipseHeight parameters of the DrawArc elementary action

18.1.1 Description

The ranges for the EllipseWidth and EllipseHeight parameters of the DrawArc elementary action (subclause 35.4) are not defined in its description.

18.1.2 Required action

Add the following sentence to the definition of the DrawArc elementary action in subclause 35.4:

"

EllipseWidth and EllipseHeight shall be equal to or greater than zero.

"

18.2 Unspecified ranges for the EllipseWidth and EllipseHeight parameters of the DrawSector elementary action

18.2.1 Description

The ranges for the EllipseWidth and EllipseHeight parameters of the DrawSector elementary action (subclause 35.4) are not defined in its description.

18.2.2 Required action

Add the following sentence to the definition of the DrawSector elementary action in subclause 35.4:

"

EllipseWidth and EllipseHeight shall be equal to or greater than zero.

"

18.3 Unspecified ranges for the EllipseWidth and EllipseHeight parameters of the DrawOval elementary action

18.3.1 Description

The ranges for the EllipseWidth and EllipseHeight parameters of the DrawOval elementary action (subclause 35.4) are not defined in its description.

18.3.2 Required action

Add the following sentence to the definition of the DrawOval elementary action in subclause 35.4:

"

EllipseWidth and EllipseHeight shall be equal to or greater than zero.

"

18.4 Wrong description in the definition of the DrawArc elementary action

18.4.1 Description

The second paragraph of the definition of the DrawArc elementary action in subclause 35.4 is wrong. It refers to the *LineColour* attribute which actually does not exist for this class. It also does not specify the line style and line width which are used to draw the object.

18.4.2 Required action

Change the second paragraph of the definition of the DrawArc elementary action in subclause 35.4 to:

"

The Arc is drawn using the current attribute of *RefLineColour*, *LineStyle*, and *LineWidth* internal attributes.

"

18.5 Missing description in the definition of the DrawPolyline elementary action

18.5.1 Description

The second paragraph of the definition of the DrawPolyline elementary action in subclause 35.4 does not specify the line colour, line style, and line width which are used to draw the object.

18.5.2 Required action

Add the following sentence to the definition of the DrawPolyline elementary action in subclause 35.4 to:

"

The line is drawn using the current attribute of *RefLineColour*, *LineStyle*, and *LineWidth* internal attributes.

"

18.6 Missing description in the definition of the DrawRectangle elementary action

18.6.1 Description

The second paragraph of the definition of the DrawRectangle elementary action in subclause 35.4 does not specify the line colour, line style, and line width which are used to draw the object.

18.6.2 Required action

Add the following sentence to the definition of the DrawRectangle elementary action in subclause 35.4 to:

"

The line is drawn using the current attribute of *RefLineColour*, *LineStyle*, and *LineWidth* internal attributes.

"

18.7 Missing description in the definition of the DrawSector elementary action

18.7.1 Description

The second paragraph of the definition of the DrawSector elementary action in subclause 35.4 does not specify the line style and line width which are used to draw the object.

18.7.2 Required action

Change the second sentence of the definition of the DrawSector elementary action in subclause 35.4 to:

"

> Lines are drawn with *RefLineColour* using the *LineStyle*, and *LineWidth* internal attributes. The surface is filled up with *RefFillColour*.

"

18.8 Missing description in the definition of the DrawOval elementary action

18.8.1 Description

The second paragraph of the definition of the DrawOval elementary action in subclause 35.4 does not specify the line style and line width which are used to draw the object.

18.8.2 Required action

Change the second sentence of the definition of the DrawOval elementary action in subclause 35.4 to:

"

> Lines are drawn with *RefLineColour* using the *LineStyle*, and *LineWidth* internal attributes. The ellipse is filled up with *RefFillColour*.

"

18.9 Missing description in the definition of the DrawPolygon elementary action

18.9.1 Description

The second paragraph of the definition of the DrawPolygon elementary action in subclause 35.4 does not specify the line style and line width which are used to draw the object.

18.9.2 Required action

Change the second sentence of the definition of the DrawPolygon elementary action in subclause 35.4 to:

"

> Lines are drawn with *RefLineColour* using the *LineStyle*, and *LineWidth* internal attributes. The polygon is filled up with *RefFillColour*.

"

18.10 Clarification on the appearance of DynamicLineArt objects

18.10.1 Description

Rec. ITU-T T.172 | ISO/IEC 13522-5 does not specify the exact appearance of DynamicLineArt object. This must be defined in an application domain.

18.10.2 Required action

Add the following Note before subclause 35.1 of the DynamicLineArt class:

"

> NOTE – This part of Rec. ITU-T T.172 | ISO/IEC 13522-5 does not define the exact visible appearance of DynamicLineArt objects.

"

18.11 Unnecessary Restriction on the usage of SetPosition, BringToFront, SendToBack, PutBefore, and PutBehind elementary actions

18.11.1 Description

DynamicLineArt objects are unnecessarily cleared every time one of the SetPosition, BringToFront, SendToBack, PutBefore, and PutBehind elementary actions is executed.

18.11.2 Required action

Remove the restrictions for the SetPosition, BringToFront, SendToBack, PutBefore, and PutBehind elementary actions of the DynamicLineArt class in subclause 35.4.

19 Text Class

19.1 Rendering of Text content

19.1.1 Description

The direction of the text rendering is not clear, even with regard to the StartCorner and LineOrientation attributes of the Text class (subclause 36.1.2).

19.1.2 Required action

Add the following Note to the StartCorner attribute in subclause 36.1.2:

"

> NOTE – The exact interpretation of the *StartCorner* attribute and the orientation of characters is not defined by this part of Rec. ITU-T T.172 | ISO/IEC 13522-5.

"

Add the following Note to the LineOrientation attribute in subclause 36.1.2:

"

> NOTE – The exact interpretation of the *LineOrientation* attribute and the orientation of characters is not defined by this part of Rec. ITU-T T.172 | ISO/IEC 13522-5.

"

19.2 Introduction of the ContentPreparation behaviour

19.2.1 Description

A new behaviour *ContentPreparation* introduced in the Root, Ingredient, and Visible classes is also introduced for the Text class (subclause 36.3) to clarify the process of content retrieval.

19.2.2 Required action

Change the introduction text of subclause 36.3 to:

"

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

"

Add the following behaviour to the internal behaviours of the Text class:

"

ContentPreparation	Apply steps 1 to 3 of the *ContentPreparation* behaviour of the base class *synchronously*. The following steps are *asynchronous* and occur when the content of the object has been fully retrieved: 4) Update the value of the *TextData* internal attribute of the target *Text*. 5) If the object is being displayed, then display the object again with the new value of *TextData* immediately. 6) Generate a *ContentAvailable* event.

"

19.3 Removal of the SetData elementary action

19.3.1 Description

Since a new internal behaviour *ContentPreparation* is introduced for the Text class in subclause 36.3, the modified definition of the SetData elementary action can be removed.

19.3.2 Required action

Remove the definition of the SetData elementary action from subclause 36.4.

19.4 Modification of the introduction text for subclause 36.4

19.4.1 Description

Since the new definition of the SetData elementary action is removed from the Text class, the introduction to subclause 36.4 needs to be modified.

19.4.2 Required action

Change the introduction to subclause 36.4 to:

"

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

"

20 Stream Class

20.1 Spelling error in the description of the Multiplex attribute

20.1.1 Description

There is a spelling error in the last sentence of the description of the Multiplex exchanged attribute of the Stream class (subclause 37.1.2).

20.1.2 Required action

Change the last sentence of the description of the Multiplex exchanged attribute of the Stream class (subclause 37.1.2) to:

"

- Sequence of inclusions of Video, Audio and RTGraphics objects.

"

20.2 Clarification for the Multiplex exchanged attribute

20.2.1 Description

A clarification for the assignment of substreams to the objects specified in the Multiplex attribute is needed.

20.2.2 Required action

Add the following Notes to the description of the Multiplex exchanged attribute of the Stream class (subclause 37.1.2):

"

NOTE 1 – The application domain is responsible to define the means to map substreams in the encoded content to objects in the Multiplex attribute.

NOTE 2 – It is the responsibility of the application domain to define the behaviour if the type or number of substreams in the encoded content does not match the type or number of objects in the Multiplex attribute of the Stream object.

"

20.3 Spelling error in the formal syntax description for the SetCounterEndPosition elementary action

20.3.1 Description

The formal syntax description for the SetCounterEndPosition elementary action is in contradiction to Annex A due to spelling errors.

20.3.2 Required action

Replace the formal description for the SetCounterEndPosition elementary action (subclause 37.4) by the following text:

"

```
 SetCounterEndposition   -->   Target,
                               NewCounterEndPosition
Target                   -->   GenericObjectReference
NewCounterEndPosition    -->   GenericInteger
```

"

20.4 Remove unnecessary restriction from the SetCounterEndPosition elementary action

20.4.1 Description

Step 1 of the description of the SetCounterEndPosition elementary action in subclause 37.4 is an unnecessary restriction because the elementary action does not change the position, speed, or direction of the stream.

20.4.2 Required action

Remove the first step from the description of the SetCounterEndPosition elementary action in subclause 37.4.

20.5 Ambiguity between the SetCounterEndPosition elementary action and the Looping exchanged attribute

20.5.1 Description

The third part of the description of the SetCounterEndPosition elementary action in subclause 37.4 is not clear in the case if the Looping exchanged attribute is set to infinite.

20.5.2 Required action

Change the former third step (see subclause 20.4 of this corrigendum) of the description of the SetCounterEndPosition elementary action in subclause 37.4 to:

"

If the target Stream is active and *NewCounterEndPosition* is already passed, stop the target Stream. If Looping is not equal to 1, a new looping play back shall be played.

"

20.6 Introduction of the ContentPreparation behaviour

20.6.1 Description

A new behaviour *ContentPreparation* introduced in the Root, Ingredient, and Visible classes is also introduced for the Stream class (subclause 37.3) to clarify the process of content retrieval.

20.6.2 Required action

Add the following behaviour to the internal behaviours of the Stream class in subclause 37.3:

"

ContentPreparation Apply steps 1 to 3 of the *ContentPreparation* behaviour of the base class *synchronously*.

The following steps are *asynchronous* and occur when the content of the Stream is in an optimal state for playing the active *StreamComponents*:

4) If the active *StreamComponents* are being played, then play the active *StreamComponents* again with the value of *Content* immediately.

5) Generate a *ContentAvailable* event.

NOTE – The content of the Stream is in an optimal state to be played once the first individually presentable portion of the Stream content has been retrieved.

"

20.7 Missing definition for SetData on scaled Video objects

20.7.1 Description

There is no indication of what the scale of a video is when initially loaded. In the case of a SetData elementary action, it is not clear whether the scaling factor is retained or the video is reset to the original size.

20.7.2 Required action

Add the following definition to the elementary actions in subclause 37.4:

"

SetData NOTE – A SetData elementary action resets the scaling factor of a video object contained in a Stream object to its original value.

"

20.8 Clarification of the StreamPlaying and StreamStopped events

20.8.1 Description

The definition of the *StreamPlaying* and *StreamStopped* events is not clear when the events are generated in case for looping and in case of a continuous stream, e.g., in a broadcast environment.

20.8.2 Required action

Change the definitions of the *StreamPlaying* and *StreamStopped* events in subclause 37.2 to:

"

StreamPlaying	This event is generated when a Stream multiplex has started playing. More specifically, it is generated simultaneously when the stream starts playing to the user. • No associated data. NOTE 1 – Each application domain may choose to give a different meaning to that degree of stream starts, this part of Rec. ITU-T T.172 | ISO/IEC 13522-5 does not specify any meaning or time requirement in this matter. It is possible that this event never occurs in some circumstance, i.e., in a broadcast environment. NOTE 2 – During looping, the event is generated only once at the beginning of the first loop.
StreamStopped	This event is generated when a Stream multiplex has stopped playing. More specifically, it is generated as soon as the ending of a stream has been presented to the user. Note that the *RunningStatus* of the Stream object is not affected by the occurrence of a *StreamStopped* event. • No associated data. NOTE 1 – Even if the *RunningStatus* of the stream component objects (i.e., Audio, Video, RTGraphics) is set to false, this event occurs when the stream ends. NOTE 2 – The event is generated when all the loops are done.

"

20.9 Clarification

20.9.1 Description

The SetSpeed elementary action in subclause 37.4 does not have a relation to the status of the stream itself, it should only change the speed of the stream, even if the speed is zero. The elementary action should not generate *StreamPlaying* and *StreamStopped* events.

20.9.2 Required action

Add the following Note to the SetSpeed elementary action in subclause 37.4:

"

NOTE 3 – This action does not have a relation to the *RunningStatus* internal attribute of the stream contents themselves, it should only change the speed of the stream, even if the speed is zero. It shall not generate the *StreamPlaying* and *StreamStopped* events.

"

21 Audio Class

21.1 Missing activation and deactivation behaviours for the Audio class

21.1.1 Description

The *Activation* and *Deactivation* behaviours for the Audio class are missing. An Audio object is never activated. The actual presentation of the audio content is never stopped (as it is for stopping the video presentation as defined in the *Deactivation* behaviour of the Visible class).

21.1.2 Required action

Add the following *Activation* behaviour and Note to the internal behaviours of the Audio class (subclause 38.3):

"

Activation	1) Execute the *Activation* behaviour as defined in the base class.
	2) Set the *RunningStatus* attribute to True and generate an *IsRunning* event.

NOTE – The actual presentation of an Audio object is started during the *Activation* behaviour of the Stream object of which the Audio object is a component.

"

Add the following *Deactivation* behaviour to the internal behaviours of the Audio class (subclause 38.3):

"

Deactivation	1) If the *RunningStatus* attribute of the object is False; ignore the behaviour; otherwise
	2) Stop presenting the audio.
	3) Execute the *Deactivation* behaviour as defined in the base class.

"

22 Interactible Class

22.1 Setting the InteractionStatus attribute to False before deactivating an Interactible object

22.1.1 Description

The *InteractionStatus* internal attribute of an Interactible object is not reset to False when the object is deactivated, i.e., it still receives the user input.

22.1.2 Required action

Add the following *Deactivation* behaviour to subclause 41.3:

"

Deactivation Set the *InteractionStatus* internal attribute to False.

NOTE – The *InteractionStatus* internal attribute of an Interactible object needs to be set to False when the object is deactivated because the SetInteractionStatus elementary action can only be targeted to active Interactibles.

"

23 Slider Class

23.1 Missing initialization of the SliderValue internal attribute

23.1.1 Description

The default value of the *SliderValue* internal attribute is not defined.

23.1.2 Required action

Add the following definition to subclause 42.1.3:

"

SliderValue	...
	• Integer value.
	• Initial Value: InitialValue

"

23.2 Missing initialization of the Portion internal attribute

23.2.1 Description

The default value of the *Portion* internal attribute is not defined.

23.2.2 Required action

Add the following definition to subclause 42.1.3:

"

Portion ...

- Integer value.
- Initial Value: InitialPortion, only when InitialPortion is encoded

"

23.3 Forbid targeting the SetData elementary action to Slider objects

23.3.1 Description

The Slider class has no encoded *Content* attribute. To target the SetData elementary action to a Slider object should not be allowed. The description is missing from subclause 42.4.

23.3.2 Required action

Add the following definition to subclause 42.4:

"

SetData SetData shall not be targeted to Slider.

"

23.4 Clarification of the Deactivation behaviour

23.4.1 Description

Subclause 22.1 of this corrigendum adds a *Deactivation* behaviour for the Interactible class. The order in which the *Deactivation* behaviours of the base classes (i.e., the Interactible and Visible class) are executed has to be defined.

23.4.2 Required action

Add the following *Deactivation* behaviour to subclause 42.3:

"

Deactivation Execute the following sequence of actions:

1) Apply the *Deactivation* behaviour of the Interactible class.
2) Apply the *Deactivation* behaviour of the Visible class.

"

23.5 Wrong description of the Step elementary action

23.5.1 Description

The second step of the description of the *Step* elementary action (subclause 42.4) is wrong. It says:

2) If *NbOfSteps* is negative, decrease the value of *SliderValue* by *NbOfSteps* × *StepSize*.

If *NbOfSteps* is already negative then the result of the expression *NbOfSteps* × *StepSize* is a negative value, too. However, if the value of *SliderValue* is to be decreased by a negative value, then *SliderValue* will be actually increased.

23.5.2 Required action

Remove step 2 of the description of the Step elementary action (subclause 42.4) and change step 1 to:

"

Adjust the value of *SliderValue* by *NbOfSteps* × *StepSize*.

"

24 EntryField Class

24.1 Clarification of the Deactivation behaviour

24.1.1 Description

Subclause 22.1 adds a *Deactivation* behaviour for the Interactible class. The order in which the *Deactivation* behaviour of the base classes (i.e., the Interactible and Text class) are executed has to be defined.

24.1.2 Required action

Add the following *Deactivation* behaviour to subclause 43.3:

"

Deactivation Execute the following sequence of actions:

1) Apply the *Deactivation* behaviour of the Interactible class.
2) Apply the *Deactivation* behaviour of the Text class.

"

25 HyperText Class

25.1 Spelling error in the Provisions of Use of the GetLastAnchorFired elementary action

25.1.1 Description

The second part of the Provisions of Use of the GetLastAnchorFired elementary action (subclause 44.4) is misspelled.

25.1.2 Required action

Change the second part of the Provisions of Use of the GetLastAnchorFired elementary action of the HyperText class in subclause 44.4 to:

"

LastAnchorFiredVar shall refer to an active OctetStringVariable object.

"

25.2 Clarification of the Deactivation behaviour

25.2.1 Description

Subclause 22.1 adds an explicit *Deactivation* behaviour for the Interactible class. The order in which the *Deactivation* behaviour of the base classes (i.e., the Interactible and Text class) are executed has to be defined.

25.2.2 Required action

Add the following *Deactivation* behaviour to subclause 44.3:

"

Deactivation	Execute the following sequence of actions:
	1) Apply the *Deactivation* behaviour of the Interactible class.
	2) Apply the *Deactivation* behaviour of the Text class.

"

26 Button Class

26.1 Missing exclusion of the SetData elementary action for Button class objects

26.1.1 Description

The SetData elementary action is not excluded from the Button Class.

26.1.2 Required action

Add the following paragraph to subclause 45.4:

"

SetData	This action shall not be targeted to a Button object.

"

27 Hotspot Class

27.1 Missing table for the list of changed inherited attributes

27.1.1 Description

The table for the inherited attributes whose semantics has changed is missing for the Hotspot class (see also subclauses 28.1 and 29.2 of this corrigendum).

27.1.2 Required action

Add the following table for the inherited attributes whose semantics has changed for the Hotspot classes (subclause 46.1.1):

"

Attribute name	Defined in	Constraints and requirements
SelectionStatus	Button	Rendering of a Hotspot object shall depend on the *SelectionStatus* attribute. When the *SelectionStatus* attribute and the *EngineResp* are both True, the Hotspot shall be rendered in a way that signals to the user that selection has taken place. For this rendering process, the attribute *ButtonRefColour* may be used. In all other cases, the Hotspot shall have no visual rendering except such rendering as is prescribed by the *HighlightStatus* of its base class.

"

28 PushButton Class

28.1 Missing table for the list of changed inherited attributes

28.1.1 Description

The table for the inherited attributes whose semantics has changed is missing for the PushButton class (see also subclauses 27.1 and 29.2 of this corrigendum).

28.1.2 Required action

Add the following table for the inherited attributes whose semantics have changed for the PushButton classes (subclause 47.1.1):

"

Attribute name	Defined in	Constraints and requirements
SelectionStatus	Button	Rendering of a PushButton object shall depend on the *SelectionStatus* attribute. When the *SelectionStatus* attribute and the *EngineResp* attribute are both True, the *PushButton* shall be rendered in a way that signals to the user that selection has taken place. This rendering shall depict a button that has been pressed. In all other cases, the Button shall be rendered in a way that signals to the user that selection has not taken place. This rendering shall depict a button that has not been pressed.

"

28.2 Clarification for the SetLabel elementary action

28.2.1 Description

The description of the OriginalLabel attribute (subclause 47.1.2) defines that the label is a one-line piece of text. This is not repeated for the Label attribute (subclause 47.1.3) and the SetLabel elementary action (subclause 47.4).

28.2.2 Required action

Remove the word "one-line" from the definition of the OriginalLabel attribute (subclause 47.1.2). Add the following Note:

"

NOTE – How the label is rendered is not defined by this part of Rec. ITU-T T.172 | ISO/IEC 13522-5.

"

29 SwitchButton Class

29.1 Missing Provisions of Use and Syntax Description of the Select elementary action

29.1.1 Description

Both the Provisions of Use and the Syntax Description for the Select elementary action of the SwitchButton class (subclause 48.4) is missing.

29.1.2 Required action

Add the following sentence to the end of the description of the Select elementary action of the SwitchButton class in subclause 48.4:

"

The provisions of use and syntax descriptions of the action are unchanged.

"

29.2 Missing table for the list of changed inherited attributes

29.2.1 Description

The table for the inherited attributes whose semantics has changed is missing for the SwitchButton class (see also subclauses 27.1 and 28.1).

29.2.2 Required action

Add the following table for the inherited attributes whose semantics has changed for the SwitchButton class (subclause 48.1.1).

"

Attribute name	Defined in	Constraints and requirements
SelectionStatus	PushButton	Rendering of a SwitchButton object shall depend on the *SelectionStatus* attribute. When the *SelectionStatus* attribute and the *EngineResp* are both True, the SwitchButton shall be rendered in a way that signals to the user that selection has taken place. This rendering shall depict a radio button or a checkbox that has been selected, or a push button that has been pressed, depending on the *ButtonStyle* attribute. In all other cases, the Button shall be rendered in a way that signals to the user that selection has not taken place. This rendering shall depict a radio button or a checkbox that has not been selected, or a button that has not been pressed.

"

29.3 Superfluous SetLabel elementary action in SwitchButton class

29.3.1 Description

The SetLabel elementary action is defined for the PushButton as well as for the SwitchButton class without any change in the semantics. The SwitchButton class inherits from the PushButton class, therefore the definition of the separate SetLabel elementary action for the SwitchButton class is superfluous.

29.3.2 Required action

Remove the definition of the SetLabel elementary action from subclause 48.4.

30 Appendix I

30.1 Contradiction to the clarification about the use of the TransitionTo elementary action

30.1.1 Description

The last sentence of the informative description in Appendix I contains a wrong recommendation.

30.1.2 Required action

Remove the last sentence of Appendix I.

31 Appendix II

31.1 Missing definition of the default background colour of the screen

31.1.1 Description

The default background colour of the screen is not defined by this part of Rec. ITU-T T.172 | ISO/IEC 13522-5.

31.1.2 Required action

Add a new clause to Appendix II:

"

II.10 Default screen background colour

The default background colour of the screen is not defined by this part of Rec. ITU-T T.172 | ISO/IEC 13522-5.

"

31.2 Definition on how to use the ObjectInformation attribute

31.2.1 Description

A method is needed to describe the profile an engine needs to support in order to execute an interchanged application.

31.2.2 Required action

Add a new clause to Appendix II:

"

II.11 Profile information encoding

The *ObjectInformation* exchanged attribute can be used to encode further information for identifying the profile for the Application. Encoding the profile information in the *ObjectInformation* attribute is optional. Besides encoding profile information, it is possible to add other information to the *ObjectInformation* exchanged attribute at the same time.

The following definition specifies the format in which the profile information is stored in the *ObjectInformation* exchanged attribute:

```
ObjectInformation ::= ProfileInfo ArbitraryInfo .

ProfileInfo       ::= '#' ProfileItem+ '#' .

ProfileItem       ::= PIContent [ '.' ProfileItem ] .

PIContent         ::= STRING .

ArbitraryInfo      ::= STRING .

STRING            ::= <printable character or white space, except '.'
                      and '#']> + .
```

"

32 Appendix III

32.1.1 Description

The examples given in Rec. ITU-T T.172 | ISO/IEC 13522-5 for the handling of TokenManager, TokenGroup, and ListGroup classes are insufficient.

32.1.2 Required action

Add new Appendix III as follows:

"

Appendix III

Object grouping mechanisms

(This appendix does not form an integral part of this Recommendation)

III.1 TokenManager

The TokenManager class is an abstract mix-in class that defines functions to manage the navigation of a logical token among a group of elements. The TokenManager defines the following:

- A variable indexing the slot with the token (the *TokenPosition* attribute).
- A table of token values intended to be used as a way to describing navigation of the token from one slot to another (the *MovementTable* attribute).
- Actions to cause the token position to change (*Move* and *MoveTo*).
- Events generated upon the movement of the token (*TokenMovedFrom* and *TokenMovedTo*).

The *MoveTo* action sets the *TokenPosition* to the value of the parameter to the action, whereas the *Move* action uses the *MovementTable* to calculate the new *TokenPosition*. The *MovementTable* consists of a sequence of Movement structures. Each Movement structure encapsulates the token transitions to be made on a certain type of movement. Typically, each Movement corresponds to a navigation direction, such as "up" or "down". The parameter to the *Move* action is used to select a particular Movement structure. Each Movement is a sequence of Integers. The value of *TokenPosition* is used to index a particular index in the Movement structure.

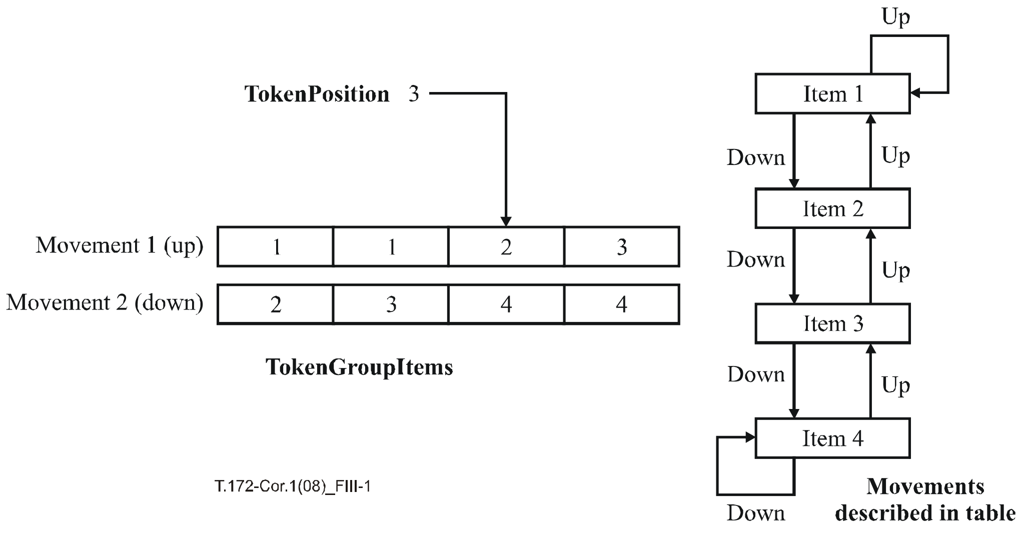


Figure III.1 – The MovementTable in TokenManager

Using the *MovementTable* above, if the *TokenPosition* is 3, then the action Move(2) will access the third item in Movement structure 2, and cause a token transition from 3 to 4. If however, the *TokenPosition* was 2, then Move(2) would cause the token to move from 2 to 3.

When used in TokenGroup, the token maps into the Visibles defined in the *TokenGroupItems* attribute. When used in a ListGroup, the token maps into the cells defined by the *Positions* attribute. Therefore, in a TokenGroup, the number of *Visibles* defined in *TokenGroupItems* shall equal the number of entries in each Movement structure in the *MovementTable* attribute (if encoded). Additionally, in a ListGroup, the number of cells defined by the *Positions* attribute shall equal the number of entries in each Movement structure in the *MovementTable* attribute (if encoded).

III.2 TokenGroup

The TokenGroup class uses the facilities of TokenManager to logically group a set of Visibles. The Visibles are declared in the *TokenGroupItems* attributes and do not change during the lifecycle of the TokenGroup. The token is used to direct actions to a particular member of the TokenGroup.

Each entry in the *TokenGroupItems* may contain a sequence of ActionSlots, each of which may contain a sequence of Action objects. When the *CallActionSlot* action occurs, the ActionSlots of the item with the token is accessed. The ActionSlot with the index matching the parameter to *CallActionSlot* is executed.

This mechanism is convenient for directing events to one of a set of targets, for example, in menus.

Example of a TokenGroup with 4 items:

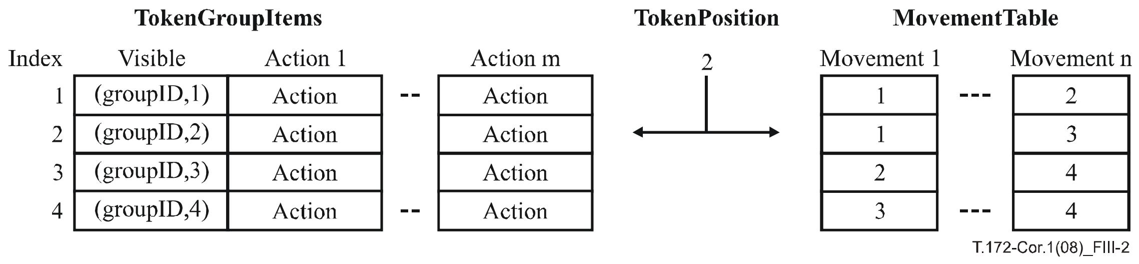


Figure III.2 – Data structures in TokenGroup

III.3 ListGroup

The ListGroup class inherits the behaviour of TokenGroup, but instead of handling a fixed set of Visibles, it handles a dynamically changing list of Visibles, which are mapped into a set of display positions (or cells). The facilities inherited from TokenGroup are used to manage the set of cells. ListGroup defines the new *ItemList* attribute to hold the actual list of Visibles, and defines a new set of actions for handling this list.

The cells are constructed from the *Positions* attribute, the *MovementTable* attribute from TokenManager, and the ActionSlots from TokenGroup. The Visibles defined in *TokenGroupItems* are used to initialize the *ItemList*, but are not used whilst the ListGroup is active. Figure III.3 below shows this behaviour. Note that duplicate items in the Visibles list are removed. Also, Null ObjectReferences are not added to the *ItemList*. This permits action slots to be declared even when there is no initial content in the list.

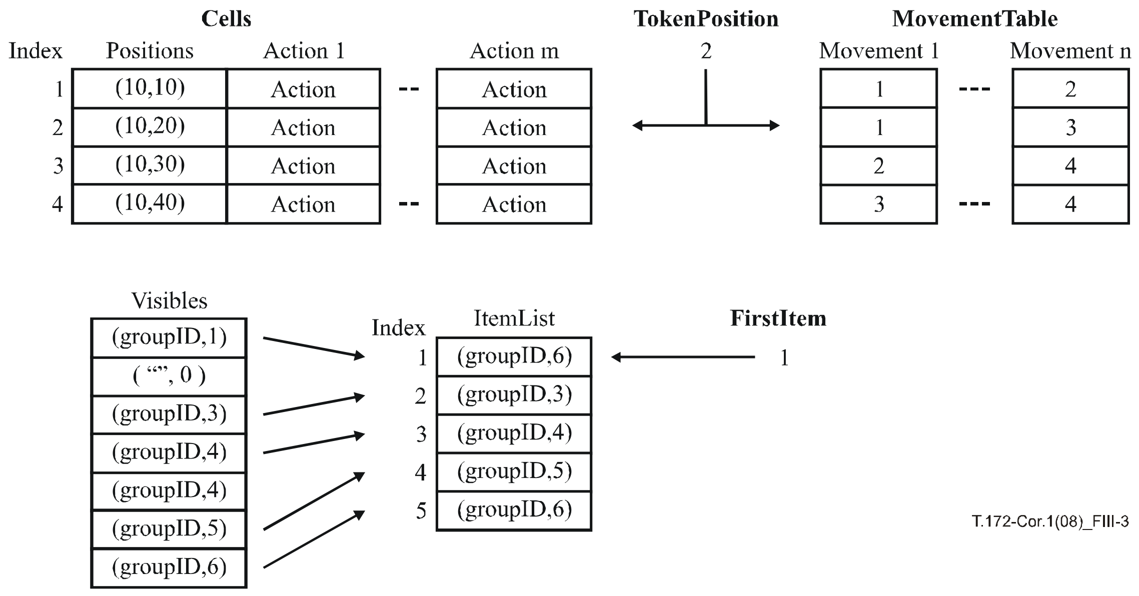


Figure III.3 – Preparation of ListGroup data structures

The Visibles in the *ItemList* are mapped onto the cells by defining a 'window' on the list of items. The window is the same size as the number of cells. The *FirstItem* attribute defines the index position in the *ItemList* of the item that will be presented at the top of the windows, and therefore in the first cell. Items in the 'window' are moved to the coordinates defined by the *Positions* attributes and are activated. Those items not in the window are moved to their original positions and deactivated.

The presentation of items in the list also depends on the value of the *WrapAround* attribute. If there are at least as many items in the *ItemList* as there are cells, the window can be visualized easily. This situation is depicted twice below. In the first example, *WrapAround* is false, i.e., the list is not wrapping:

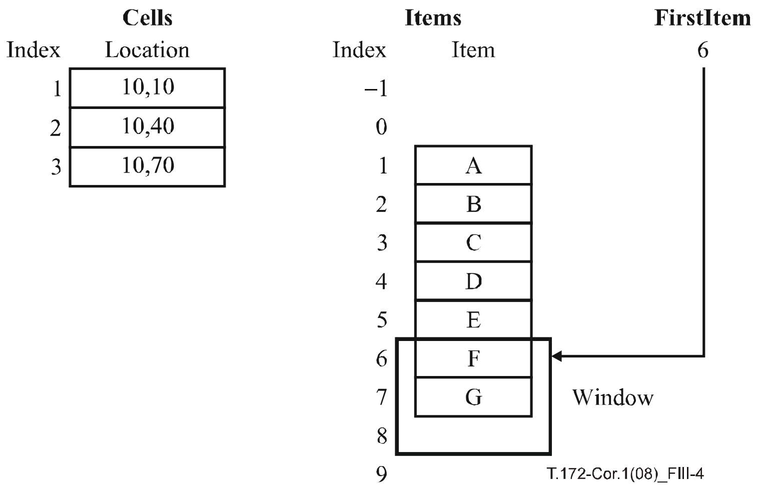


Figure III.4 – Presentation when WrapAround is False

In this case, items F and G are presented at cell 1 and 2 respectively, while the other items are not presented. Cell 3 remains unused.

In the next example, *WrapAround* is true, i.e., the list is wrapping:

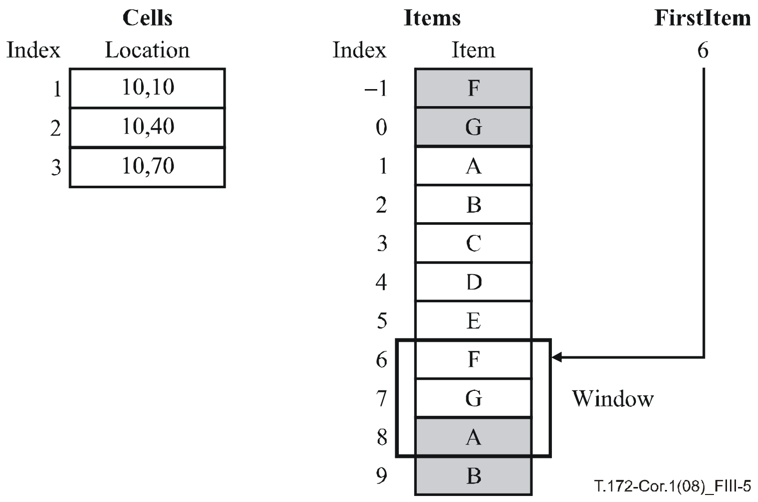


Figure III.5 – Presentation when WrapAround is True

In this case, items F, G and A are presented at cell 1, 2 and 3 respectively, while the other items are not presented. Note that in a wrapping list, the indices of the items are extended (modulo the number of items in the list). For instance, ...,–4, 3, 10, ... are all valid indices for item C in the above example.

The other situation occurs if there are less items in the *ItemList* as there are cells. This situation is depicted twice below. In this next example, *WrapAround* is false, i.e., the list is not wrapping:

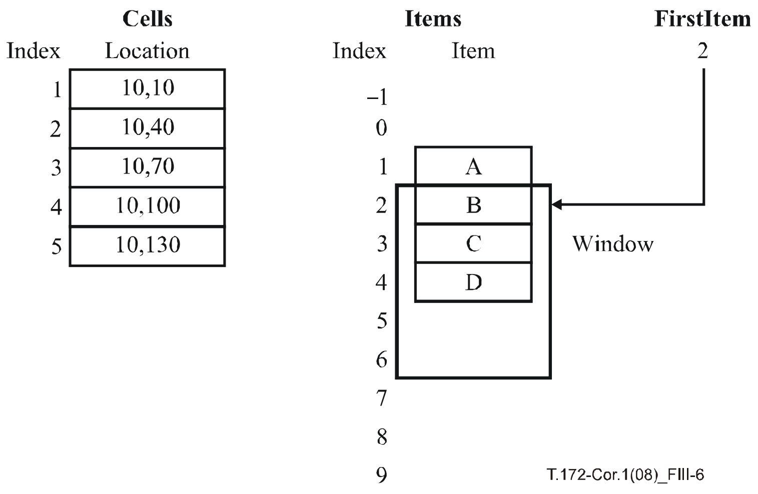


Figure III.6 – Presentation of few elements when WrapAround is False

This case is much like the first example only the window is bigger. Items B, C and D are presented at cell 1, 2 and 3 respectively, while item A is not presented. Cells 4 and 5 remain unused.

In the next example, *WrapAround* is true, i.e., the list is wrapping:

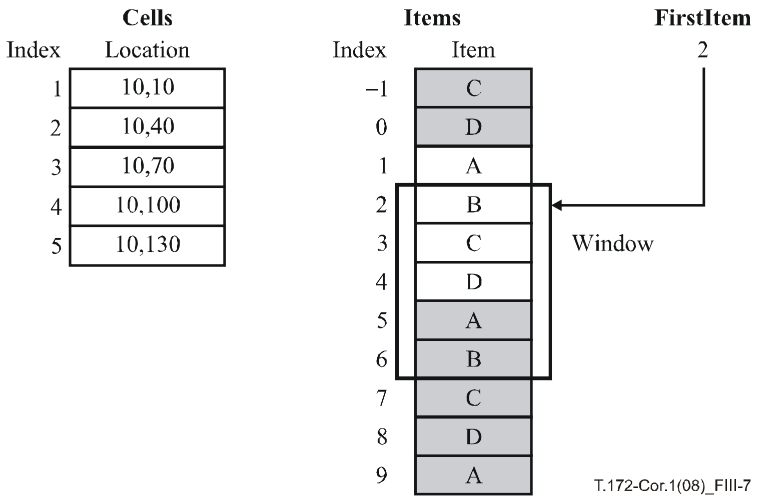


Figure III.7 – Presentation of few elements when WrapAround is True

The important thing in this case is that there are two candidate cells to present item B. Since Visibles can have just one position, items can also just be presented in one cell. If, according to the window, an item can be presented at more than one cell, it will be presented only at the cell with the lowest index. In this case, item B will be presented at cell 1. Items C, D and A are presented at cell 2, 3 and 4 respectively. Cell 5 remains unused. Note that, when the number of items is less than the number of cells, cells will remain unused, even if the list is wrapping.

Cells may at any time be 'unused', meaning that there may be no item mapped into that cell. All TokenGroup facilities are unaffected by this situation – the token may be held by an unused cell, and because ActionSlots are attached to the cell position and not a Visible, the CallActionSlot action operates as normal even if the cell is empty. The GetCellItem action will place the Null ObjectReference in the *ObjectRefVariable* if the cell is unused. The author should be aware that attempting to use a Null ObjectReference could lead to unexpected results. The author is strongly advised to test against the Null ObjectReference value before accessing the contents of cells.

"

33 Other parts

This clause describes other required changes to this part of Rec. ITU-T T.172 | ISO/IEC 13522-5 which are not related to the defined classes.

33.1 Clarification for the use of "null" and "none" in clause 3

33.1.1 Description

Rec. ITU-T T.172 | ISO/IEC 13522-5 uses both the words "null" and "none". These words are intended to be synonymous, but this is not stated in Rec. ITU-T T.172 | ISO/IEC 13522-5.

33.1.2 Required action

Add the following description after subclause 3.24:

"

Null

In this text, *Null* and *None* values are synonymous.

"

33.2 Wrong description of the OnStartUp attribute subclause 5.3

33.2.1 Description

The description for the OnStartUp attribute in the second paragraph of subclause 5.3 is wrong.

33.2.2 Required action

Change the second sentence of the second paragraph of subclause 5.3 to:

"

When the Application object becomes active, it automatically executes the OnStartUp action.

"

33.3 Misleading description of Link event data in subclause 5.6

33.3.1 Description

The text describing the conditions under which a link fires does not take into account that the EventData is not encoded.

33.3.2 Required action

Replace the second last sentence ("In other words, a Link fires ...") in subclause 5.6 with the following text:

"

In other words, a Link fires only if it is active and only if the right event is generated by the right object. If the event data of the link is encoded, the link also requires the right event parameter in order to fire.

"

33.4 Wrong OMT diagram of the Presentable class in subclause 5.11

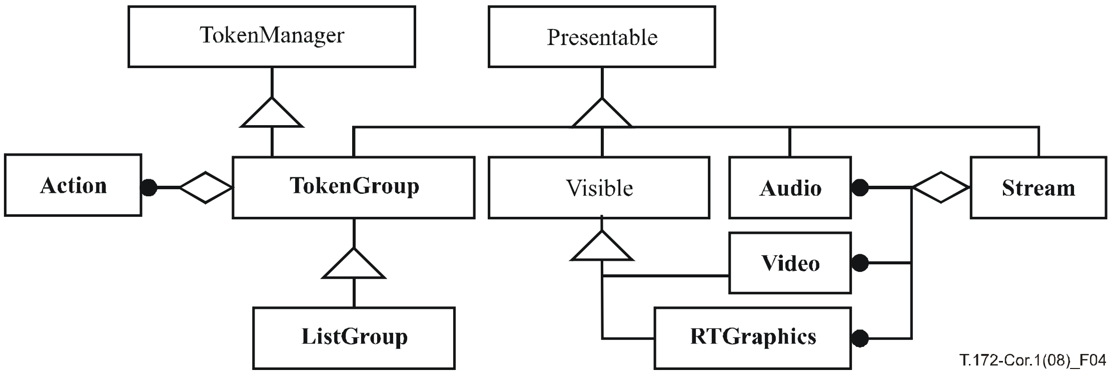
33.4.1 Description

The inheritance of the Action class in the OMT diagram in subclause 5.11 is wrong. The Action class does not inherit from the Presentable class.

33.4.2 Required action

Remove the inheritance of the Actions class from the Presentable class. Replace Figure 4 in subclause 5.11 with the following:

"



"

33.5 Definition of Context Change

33.5.1 Description

The definition of a Context Change is missing in clause 3.

33.5.2 Required action

Add the following definition to clause 3:

"

> **context change:** The point of transition between one group object completing its destruction behaviour and another group beginning its preparation behaviour.

"

33.6 Definition of Presented

33.6.1 Description

A definition for the term *Presented* is missing in clause 3.

33.6.2 Required action

Add the following definition to clause 3:

"

> **presented:** "Presented" means that the presentable object is completely rendered on the screen. In case of streams, the meaning of "completely rendered" is that the presentation has been started.

"

33.7 Clarification for contradictions in the formal syntax description for actions and objects in the text and the Annex

33.7.1 Description

A clarification is needed which states that the syntax definition in the annex is the normative syntax description in case of contradictions.

33.7.2 Required action

Add the following text to subclause 7.5:

"

If there is a contradiction between the formal syntax definition in the class descriptions and Annexes A and B, the annex descriptions are normative.

"

33.8 Missing definition of the Null ObjectReference

33.8.1 Description

The Null ObjectReference is used within the MHEG-5 standard, but not defined.

33.8.2 Required action

Add the following text to the end of subclause 50.1:

"

The Null ObjectReference consists of a zero-length OctetString for the GroupIdentifier and an ObjectNumber of zero.

"

33.9 Missing definition of the Null ContentReference

33.9.1 Description

The Null ContentReference is used within the MHEG-5 standard, but not defined.

33.9.2 Required action

Add the following text to the end of subclause 50.2:

"

The Null ContentReference consists of a zero-length OctetString.

"

33.10 Clarification of the values of ObjectRefVariable objects

33.10.1 Description

A value of an ObjectRefVariable object can be any ObjectReference. It is not required that this ObjectReference comply with the restrictions defined in clause 51 as long as it is not used. The restrictions of clause 51 only apply at the moment the ObjectRefVariable object gets used to dereference an Object in a parameter of an elementary action.

33.10.2 Required action

Add the following paragraph to the end of clause 51:

"

The value of an ObjectRefVariable object can be any ObjectReference. It is not required that this ObjectReference comply with the restrictions defined in this clause as long as it is not used. The restrictions of this clause only apply at the moment the ObjectRefVariable object gets used to dereference an MHEG-5 object in a parameter of an elementary action.

"

33.11 Clarification for the event source when the event origin is not available

33.11.1 Description

A clarification is needed which states that the event source is defined also when the event origin is no longer available.

33.11.2 Required action

Add the following text to the end of the second paragraph of subclause 53.1:

"

The event source always uniquely identifies the object that generates the event, even when the object is no longer available.

"

33.12 Deactivation of Link objects during the execution of their LinkEffect

33.12.1 Description

The behaviour of the effect of deactivating a Link object during the execution of its LinkEffect is not very clearly stated in subclause 53.3.

33.12.2 Required action

The last paragraph of subclause 53.3 shall read as follows:

"

The Link is deactivated immediately but all remaining elementary actions of the LinkEffect remain queued in the execution stack of the MHEG-5 engine.

"

33.13 Change of Variable objects during the asynchronous execution of a Program

33.13.1 Description

The Note in the definition before the Provisions of Use in the description of the Fork elementary action (subclause 14.4) defines that an MHEG-5 application must neither access the Variable object referenced by *ForkSucceeded* nor those Variables of *Parameters* which are output parameters before the *AsynchStopped* event has been dealt with.

33.13.2 Required action

Add the following paragraph to the end of subclause 53.3:

"

Variables can only be updated in-between the execution of the LinkEffect of two Link objects which have been triggered by asynchronous events.

"

33.14 Removing of asynchronous events from the event queue during context change

33.14.1 Description

The current description in subclause 53.3 does not handle events generated by the Application class and shared Ingredient objects during context changes correctly. Those events are lost during a context change.

33.14.2 Required action

Change the sixth paragraph of subclause 53.3 to:

"

Actions that change the context of the current action processing will influence both the queue of asynchronous events and the queue of actions waiting for processing. The context is changed by "TransitionTo", "Launch", "Spawn" and "Quit". If such an action occurs and is executed successfully, all pending events that are not in the scope of the new context shall be removed from the asynchronous event queue. Also, elementary actions waiting for execution shall be removed from the action queue. For example, if the TransitionTo elementary action is executed successfully, thus changing the Scene context, all asynchronous events generated by the previous Scene or by objects contained within the previous Scene, are removed. But, events created by the current Application, or shared objects contained within this Application, are not to be removed because the Application context has not changed and these events are still within scope.

"

33.15 Mapping of ComponentTags to sub-streams

33.15.1 Description

The application domain also needs to define the mapping of stream component tags to the underlying protocol which controls streaming.

33.15.2 Required action

Add another row to the table in subclause II.9:

"

MHEG-5 entity	**Mapping needed**	**Semantics of these MHEG-5 structures needs specifications**
ComponentTag	Mapping to components within stream multiplex	• in Audio, Video, and RTGraphics • DVB-SI

"

33.16 Encoding error in the Textual Notation for the LineArt class

33.16.1 Description

The second closing curly brace in subclause B.4.26 in the second clause is superfluous.

33.16.2 Required action

Remove the closing curly brace in subclause B.4.26 in the second clause.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2009