ItemIndex	-->	GenericInteger
ItemStatusVar	-->	ObjectReference

SelectItem (*ItemIndex*)

If *WrapAround* is False, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, ignore this action, otherwise:

If *WrapAround* is True, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, the *ItemIndex* shall be interpreted modulo the number of items in the *ItemList* attribute.

Apply the *Select(ItemIndex)* internal behaviour.

Provisions of use:

- The *Target* object shall be an available ListGroup object

Syntax description:

SelectItem	-->	Target, ItemIndex
Target	-->	GenericObjectReference
ItemIndex	-->	GenericInteger

DeselectItem (*ItemIndex*)

If *WrapAround* is False, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, ignore this action, otherwise:

If *WrapAround* is True, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, the *ItemIndex* shall be interpreted modulo the number of items in the *ItemList* attribute.

Apply the *Deselect(ItemIndex)* internal behaviour.

Provisions of use:

- The *Target* object shall be an available ListGroup object

Syntax description:

DeselectItem	-->	Target, ItemIndex
Target	-->	GenericObjectReference
ItemIndex	-->	GenericInteger

ToggleItem (*ItemIndex*)

If *WrapAround* is False, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, ignore this action, otherwise:

If *WrapAround* is True, and if the *ItemIndex* specifies an index smaller than 1 or greater than the number of items in the *ItemList* attribute, the *ItemIndex* shall be interpreted modulo the number of items in the *ItemList* attribute.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 101 of 253

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

Provisions of use:

- The *Target* object shall be an available ListGroup object

Syntax description:

```
ToggleItem          -->   Target,
                          ItemIndex,
Target              -->   GenericObjectReference
ItemIndex           -->   GenericInteger
```

ScrollItems (*ItemsToScroll*)

Add *ItemsToScroll* to the *FirstItem* attribute, and apply the *Update* behaviour.

Provisions of use:

- The *Target* object shall be an available ListGroup object

Syntax description:

```
ScrollItems         -->   Target,
                          ItemsToScroll
Target              -->   GenericObjectReference
ItemsToScroll       -->   GenericInteger
```

SetFirstItem (*NewFirstItem*)

Set the value of the *FirstItem* attribute to *NewFirstItem*. Apply the Update behaviour.

Provisions of use:

- The *Target* object shall be an available ListGroup object

Syntax description:

```
SetFirstItem        -->   Target,
                          NewFirstItem
Target              -->   GenericObjectReference
NewFirstItem        -->   GenericInteger
```

GetFirstItem (*FirstItemVar*)

Return the current value of the *FirstItem* attribute in the IntegerVariable referenced by *FirstItemVar*.

Provisions of use:

- The *Target* object shall be an available ListGroup object.
- *FirstItemVar* shall refer to an active IntegerVariable object.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 102 of 253

Syntax description:

```
GetFirstItem        -->   Target,
                          FirstItemVar
Target              -->   GenericObjectReference
FirstItemVar        -->   ObjectReference
```

GetListSize (*SizeVar*) Return the number of items in the *ItemList* in the IntegerVariable referenced by *SizeVar*.

Provisions of use:

- The *Target* object shall be an available ListGroup object.
- *SizeVar* shall refer to an active IntegerVariable object.

Syntax description:

```
GetListSize         -->   Target,
                          SizeVar
Target              -->   GenericObjectReference
SizeVar             -->   ObjectReference
```

30.5 Formal description

```
ListGroup Class     -->   TokenGroup Class,
                          Positions,
                          WrapAround?,
                          MultipleSelection?
Positions           -->   Position*
Position            -->   XYposition,
WrapAround          -->   BOOLEAN
MultipleSelection   -->   BOOLEAN
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 103 of 253

31 Visible Class

Description	Defines the behaviour of Presentables that have a visual representation on the screen.
Base class	Presentable
Subclasses	Video, RTGraphics, Bitmap, LineArt, Text, Slider, Button
Status	Abstract class

31.1 Attributes

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

31.1.1 Inherited attributes

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

31.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

OriginalBoxSize

Original size of the bounding box of the Visible object with respect to the coordinate system of the Scene. Size is given by a pair of Integers representing respectively the width (length along X-axis) and height (length along Y-axis) of the box.

Negative values shall not be used. Values exceeding the scene coordinate system are allowed; in that case only the part of the Visible within the scene coordinate system shall be rendered.

- Pair of Integers (XBoxSize, YBoxSize).

OriginalPosition

Original position of the top left corner of the Visible with respect to the coordinate system of the Scene.

This is a set of (X, Y) co-ordinates expressing the original position of the top left corner of the visible in the coordinate system of the currently active Scene.

Negative values and values exceeding the scene coordinate system are allowed; in that case only the part of the Visible within the scene coordinate system shall be rendered.

- Optional pair of Integers (XPosition, YPosition).
- Default value: (0,0).

OriginalPaletteRef

Indicate the initial Palette object that shall be used to render colours of the Visible object.

- Optional reference to a Palette object.
- Default value: No palette.

This attribute is mandatory when other colour attributes are set and need Palette references.

31.1.3 Own internal attributes

This class defines the following additional internal attributes:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 104 of 253

BoxSize Size of the bounding box of the Visible object with respect to the coordinate system of the Scene. Parts of the Visible that fall outside the borders of the bounding box shall not be rendered. If the Visible does not completely cover the contents of the bounding box, then the remaining parts of the bounding box shall be transparent.

- Pair of Integers (XBoxSize, YBoxSize).
- Initial value: Value of the OriginalBoxSize attribute.

Position Position of the top left corner of the Visible with respect to the coordinate system of the Scene. The Visible, when rendered, shall have its top left corner at this position.

- Pair of Integers (XPosition, YPosition).
- Initial value: Value of the OriginalPosition attribute.

PaletteRef If a Palette is used, reference to the Palette object that shall be used to render colours of the Visible object.

The *Palette* attribute shall not be defined for classes specifying that OriginalPaletteRef shall not be encoded.

- Optional reference to a Palette object.
- Initial value: Value of the OriginalPaletteRef attribute.

31.2 Events

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

31.3 Internal behaviours

This class has the same internal behaviours as its base class, with the following changes in semantics:

Preparation Execute the following sequence of actions:

1. Execute steps 1, 2 and 3 of the *Preparation* behaviour as defined in the Root class.
2. If the Visible object is not referenced in the *DisplayStack* of the active Application object, add a reference to this Visible at the top of the *DisplayStack*.
3. Execute steps 4, 5 and 6 of the *Preparation* behaviour as defined in the Root class.

Destruction Execute the following sequence of actions:

1. If the Visible object is referenced in the *DisplayStack* of the active Application object, remove the reference to this Visible from the *DisplayStack*.
2. Execute the *Destruction* behaviour as defined in the base class.

Activation Execute the following sequence of actions:

1. Execute the *Activation* behaviour as defined in the base class.
2. Display the Visible according to its position in the *DisplayStack* and to the position and the bounding box defined by the *Position* and *BoxSize* attributes.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 105 of 253

3. Set the *RunningStatus* to True and generate an *IsRunning* event.

Deactivation Execute the following sequence of actions:

1. If the RunningStatus attribute of the object is False, ignore the behaviour, otherwise:
2. Stop displaying the Visible.
3. Execute *Deactivation* behaviour as defined in the base class.

31.4 Effect of MHEG-5 actions

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

SetPosition (*NewXPosition*, *NewYPosition*)

Change the location of the target Visible.

Execute the following sequence of actions:

1. Set the *Position* attribute according to *NewXPosition* and *NewYPosition*.
2. If the Visible object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available Visible object.
- *NewXPosition* and *NewYPosition* shall correspond to a location interpreted in the Scene coordinate system defined by the SceneCoordinateSystem attribute of the currently active Scene.

Syntax description:

```
SetPosition          -->   Target,
                           NewXPosition,
                           NewYPosition
Target               -->   GenericObjectReference
NewXPosition         -->   GenericInteger
NewYPosition         -->   GenericInteger
```

GetPosition (*XPositionVar*, *YPositionVar*)

Returns the location of the target Visible.

Set the Variables referenced by *XPositionVar* and *YPositionVar* to the value of the X and Y position of the target visible respectively.

Provisions of use:

- The *Target* object shall be an available Visible object.
- *XPositionVar* and *YPositionVar* shall refer to active IntegerVariable objects.

Syntax description:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 106 of 253

```
GetPosition          --> Target,
                         XPositionVar,
                         YPositionVar
Target                 > GenericObjectReference
XPositionVar         --> ObjectReference
YPositionVar         --> ObjectReference
```

SetBoxSize (*XNewBoxSize*, *YNewBoxSize*)

Change the size of the bounding box of the target Visible.

Execute the following sequence of actions:

1. Set the *BoxSize* attribute.
2. Redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available Visible object.
- *XNewBoxSize* and *YNewBoxSize* shall be positive values and different from 0.

Syntax description:

```
SetBoxSize           --> Target,
                         XNewBoxSize,
                         YNewBoxSize
Target               --> GenericObjectReference
XNewBoxSize          --> GenericInteger
YNewBoxSize          --> GenericInteger
```

GetBoxSize (*XBoxSizeVar*, *YBoxSizeVar*)

Returns the size of the bounding box of the target Visible.

Set the Variables referenced by *XBoxSizeVar* and *YBoxSizeVar* to the value of the X and Y position of the target visible respectively.

Provisions of use:

- The *Target* object shall be an available Visible object.
- *XBoxSizeVar* and *YBoxSizeVar* shall refer to active IntegerVariable objects.

Syntax description:

```
GetBoxSize           --> Target,
                         XBoxSizeVar,
                         YBoxSizeVar
Target               --> GenericObjectReference
XBoxSizeVar          --> ObjectReference
YBoxSizeVar          --> ObjectReference
```

BringToFront

Put a Visible at the foreground of the screen.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 107 of 253

Execute the following sequence of actions:

1. If the target Visible object is not referenced in the *DisplayStack* of the active Application object, ignore this action.
2. If the target Visible object is referenced in the *DisplayStack* of the active Application object, move the reference to this Visible at the top of the *DisplayStack*.
3. If the target Visible object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available Visible object.

Syntax description:

```
BringToFront        -->   Target
Target              -->   GenericObjectReference
```

SendToBack

Put a Visible at the background of the screen.

Execute the following sequence of actions:

1. If the target Visible object is not referenced in the *DisplayStack* of the active Application object, ignore this action
2. If the target Visible object is referenced in the *DisplayStack* of the active Application object, move the reference to this Visible at the bottom of the *DisplayStack*.
3. If the target Visible object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available Visible object.

Syntax description:

```
SendToBack          -->   Target
Target              -->   GenericObjectReference
```

PutBefore (*ReferenceVisible*)

Put a Visible exactly in front of another Visible in the display stack.

Execute the following sequence of actions:

1. If the target Visible object is not referenced in the *DisplayStack* of the active Application object, ignore this action.
2. If the target Visible object is referenced in the *DisplayStack* of the active Application object, move the reference to this Visible at the position exactly before the reference to the *ReferenceVisible* in the *DisplayStack*.
3. If the target Visible object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 108 of 253

active Application object.

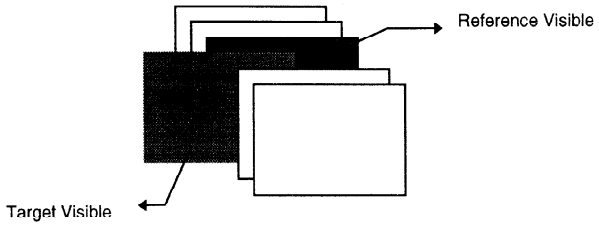


Figure 10 : Effect of PutBefore action

Provisions of use:

- The *Target* object shall be an available Visible object.
- *ReferenceVisible* shall be a reference to an available Visible object.
- The *DisplayStack* of the active Application object shall contain a reference to the *ReferenceVisible* object.

Syntax description:

```
PutBefore            -->   Target,
                           ReferenceVisible
Target               -->   GenericObjectReference
ReferenceVisible     -->   GenericObjectReference
```

PutBehind (ReferenceVisible)

Put a Visible exactly below another Visible in the display stack.

Execute the following sequence of actions:

1. If the target Visible object is not referenced in the *DisplayStack* of the active Application object, ignore this action.
2. If the target Visible object is referenced in the *DisplayStack* of the active Application object, move the reference to this Visible at the position exactly after the reference to the *ReferenceVisible* in the *DisplayStack*.
3. If the target Visible object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the *BoxSize* and *Position* attributes and according to its position in the *DisplayStack* of the active Application object.

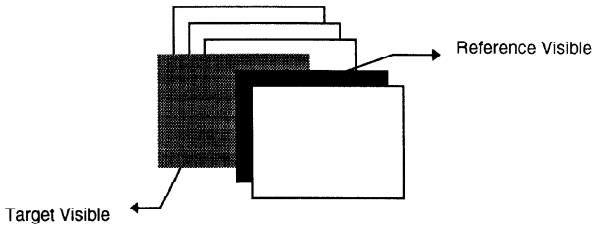


Figure 11 : Effect of PutBehind action

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 109 of 253

Provisions of use:

- The *Target* object shall be an available Visible object.
- *ReferenceVisible* shall be set or refer to a reference to an available Visible object.
- The *DisplayStack* of the active Application object shall contain a reference to the *ReferenceVisible* object.

Syntax description:

```
PutBehind           -->   Target,
                          ReferenceVisible
Target              -->   GenericObjectReference
ReferenceVisible    -->   GenericObjectReference
```

SetPaletteRef (*NewPaletteRef*)

Change the colour look-up table used to render colours of the Visible object.

Execute the following sequence of actions:

1. Set *PaletteRef* of the target Visible to *NewPaletteRef*.
2. If the target Visible is active, redraw the Visible by taking into account the new value of *PaletteRef*.

Provisions of use:

- The *Target* object shall be an available Visible object.
- *NewPaletteRef* shall contain a reference to an active Palette object.

Syntax description:

```
SetPaletteRef       --> Target,
                        NewPaletteRef
Target              --> GenericObjectReference
NewPaletteRef       --> GenericObjectReference
```

31.5 Formal description

```
Visible Class           --> Presentable class,
                            OriginalBoxSize,
                            OriginalPosition?,
                            OriginalPaletteRef?
OriginalBoxSize         --> XLength, YLength
XLength                 --> INTEGER
YLength                 --> INTEGER
OriginalPosition        --> XYPosition
OriginalPaletteRef      --> ObjectReference
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 110 of 253

32 Bitmap Class

Description	Defines the behaviour of a two-dimensional array of pixels.
Base class	Visible
Subclasses	None
Status	Concrete class

32.1 Attributes

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

32.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	Either this attribute or its corresponding default attribute in the Application class (BitmapContentHook) is mandatory for this class.
OriginalContent	Ingredient	This attribute is mandatory for this class.

32.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

Tiling — When true, this attribute indicates that the bitmap should be replicated (tiled) in the available *BoxSize* of the bitmap.

- Optional Boolean.
- Default value: False.

Original-Transparency — This attribute defines the initial transparency of those pixels of the bitmap that are marked as being transparent.

In cases where the coded representation of the bitmap itself defines the value of the Transparency, the transparency defined by content encoding and the transparency defined by this attribute should be combined. The exact algorithm is not defined by this part of ISO/IEC 13522.

- Optional Integer in the range [0,100] (percent).
- Default value: 0%.

32.1.3 Own internal attributes

This class defines the following additional internal attributes:

Transparency — Defines the transparency of those pixels of the bitmap that are marked as being transparent.

- Optional Integer in the range [0,100] (percent).
- Initial value: Value of the OriginalTransparency attribute.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 111 of 253

32.2 Events

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

32.3 Internal behaviours

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

32.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with identical semantics.

In addition, the following applicable MHEG-5 actions are defined:

ScaleBitmap (*XScale, YScale*)

If the MHEG-5 engine implements the Scaling option, the effect of this action is to scale the contents of the Bitmap to the size (*XScale, YScale*). Engines that do not implement the scaling option shall ignore this action.

Note that this action does not affect the *BoxSize* internal attribute of the Bitmap object; in other words, the Bitmap is scaled, but its bounding box remains the same.

Provisions of use:

- The *Target* object shall be an available Bitmap object.
- *XScale* and *YScale* shall be positive Integers.

Syntax definition:

```
ScaleBitmap          -->  Target,
                          XScale, YScale
Target               -->  GenericObjectReference
XScale, YScale       -->  GenericInteger
```

SetTransparency (*NewTransparency*)

Execute the following sequence of actions:

1. Change the value of the *Transparency* attribute.
2. If the bitmap is active, redraw it using the new value of the *Transparency* attribute, according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available Bitmap object.
- *NewTransparency* is set within the range [0, 100].

Syntax definition:

```
SetTransparency      -->  Target,
                          NewTransparency
Target               -->  GenericObjectReference
NewTransparency      -->  GenericInteger
```

32.5 Formal description

```
Bitmap Class              -->  Visible Class,
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 112 of 253

```
                                Tiling?,
                                OriginalTransparency?
Tiling                    -->   BOOLEAN
OriginalTransparency      -->   INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 113 of 253

33 LineArt Class

Description	Defines functionality associated with vectorial representation of graphical objects.
Base class	Visible
Subclasses	Rectangle, DynamicLineArt
Status	Concrete class

33.1 Attributes

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

33.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	Either this attribute or its corresponding default attribute in the Application class (LineArtContentHook) is mandatory for this class.
OriginalContent	Ingredient	This attribute is mandatory for this class.

33.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

BorderedBoundingBox

The BorderedBoundingBox attribute determines whether the bounding box - defined by BoxSize and Position attributes - is bordered by lines or not. When this attribute is set to True, all further drawings shall take place inside the border and shall not be drawn on top of it. OriginalLineWidth, OriginalLineStyle and OriginalRefLineColour shall be used for the rendering of the bounding box.

- Optional Boolean attribute.
- Default value: True.

OriginalLineWidth

The OriginalLineWidth attribute determines the original line width of the graphic object including the bounding box.

The OriginalLineWidth attribute is expressed in pixels in the scene coordinate space. It is specified in pixel height for horizontal lines and in pixel width for vertical lines.

The actual rendering resolution and accuracy of the MHEG-5 engine is outside the scope of this part of ISO/IEC 13522.

- Optional Integer attribute.
- Default Value: 1.

OriginalLineStyle

The OriginalLineStyle attribute determines the pattern used for the rendering of the lines of the LineArt object.

- Optional Integer attribute - one of 1, 2 or 3;

  1 means solid, 2 means dashed, 3 means dotted.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 114 of 253

- Default value: 1 (solid).

OriginalRefLine-Colour — Initial reference colour for the lines used to render the LineArt object.

The OriginalRefLineColour attribute value is expressed either as a string representing an absolute colour value or as an Integer representing a zero-based index in a colour look-up table. In the latter case, the OriginalPaletteRef attribute must contain a reference to a Palette object that is then used to translate the index to an actual colour value.

The encoding of absolute colour values as well as the actual colour resolution in the rendering process are outside the scope of this part of ISO/IEC 13522.

- Optional attribute.
- Default value: «black».

  The exact appearance on screen of default value «black» is not fully specified by this part of this part of ISO/IEC 13522.
- OctetString or Integer.

OriginalRefFillColour — Initial reference colour for the inside of a closed LineArt object.

The OriginalRefFillColour attribute value is expressed either as a string representing an absolute colour value or as an Integer representing a zero-based index in a colour look-up table. In the latter case, the OriginalPaletteRef attribute must contain a reference to a Palette object that is then used to translate the index to an actual colour value.

If the OriginalRefFillColour is not encoded, a LineArt object that represents a closed shape shall be rendered with a transparent fill colour.

The encoding of absolute colour values as well as the actual colour resolution in the rendering process are outside the scope of this part of ISO/IEC 13522.

- Optional attribute.
- Default value: «transparent».
- OctetString or Integer.

33.1.3 Own internal attributes

This class defines the following additional internal attributes:

LineWidth — The *LineWidth* attribute determines the line width of the graphical objects that are drawn by the graphical actions.

- Integer.
- Initial Value: OriginalLineWidth.

LineStyle — Width of the line that the LineArt object is intended to be rendered with.

- One of 1, 2 or 3;

  1 means solid, 2 means dashed, 3 means dotted.
- Initial Value: OriginalLineStyle.

RefLineColour — Reference colour for the lines used to render the LineArt object.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 115 of 253

- OctetString or Integer.
- Initial Value: OriginalRefLineColour.

RefFillColour Reference colour for the background colour used to render the LineArt object.

- OctetString or Integer.
- Initial Value: OriginalRefFillColour.

33.2 Events

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

33.3 Internal behaviours

This class defines no additional internal attribute.

33.4 Effect of MHEG-5 actions

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

SetLineWidth (*NewLineWidth*) Change the width of lines of a LineArt object.

Execute the following sequence of actions:

1. Set the value of the LineWidth attribute to *NewLineWidth*.
2. If the target LineArt object is active, redraw immediately the target object by taking into account the new value of the LineWidth attribute and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available LineArt object.
- *NewLineWidth* shall be set or refer to a positive integer value.

Syntax description:

```
SetLineWidth        -->   Target,
                          NewLineWidth
Target              -->   GenericObjectReference
NewLineWidth        -->   GenericInteger
```

SetLineStyle (*NewLineStyle*) Change the line style of a LineArt object.

Execute the following sequence of actions:

1. Set the value of the LineStyle attribute to *NewLineStyle*.
2. If the target LineArt object is active, redraw immediately the target object by taking into account the new value of the LineStyle attribute and according to its position in the *DisplayStack* of the active Application object.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 116 of 253

Provisions of use:

- The *Target* object shall be an available LineArt object.
- *NewLineStyle* shall be set or refer to a positive integer value: 1, 2 or 3.

Syntax description:

```
SetLineStyle        -->   Target,
                          NewLineStyle
Target              -->   GenericObjectReference
NewLineStyle        -->   GenericInteger
```

SetLineColour (*NewColour*)

Change the colour of lines of a LineArt object.

Execute the following sequence of actions:

1. Set the value of the RefLineColour attribute to *NewColour*.
2. If the target object is active, redraw immediately the target object by taking into account the new value of the LineWidth attribute and according to its position in the *DisplayStack* of the active Application object.

NewColour may be either an absolute colour value or the zero-based index of a colour in the look-up table referenced by the *PaletteRef* attribute.

Provisions of use:

- The *Target* object shall be an available LineArt object.
- If RefLineColour is currently set to an absolute colour value, *NewColour* shall be set or refer to an absolute colour value.
- If RefLineColour is currently set to an index of a colour look-up table, *NewColour* shall be set or refer to an index of a colour look-up table.

Syntax description:

```
SetLineColour       -->   Target,
                          NewColour
Target              -->   GenericObjectReference
NewColour           -->   NewColourIndex | NewAbsoluteColour
NewColourIndex      -->   GenericInteger
NewAbsoluteColour   -->   GenericOctetString
```

SetFillColour (*NewColour*)

Change the fill-in colour of a LineArt object.

Execute the following sequence of actions:

1. Set the value of the RefFillColour attribute to *NewColour*.
2. If the target object is active, redraw immediately the target object by taking into account the new value of the RefFillColour attribute and according to its position in the *DisplayStack* of the active Application object.

NewColour may be either an absolute colour value or the zero-based index of a colour in the look-up table referenced by the *PaletteRef* attribute.

If *NewColour* is not encoded, the RefFillColour attribute shall be set to «transparent».

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 117 of 253

Provisions of use:

- The *Target* object shall be an available LineArt object.
- If *RefFillColour* is not currently set, *NewColour* shall be set or refer to an absolute colour value.
- If RefFillColour is currently set to an absolute colour value, *NewColour* shall be set or refer to an absolute colour value.
- If RefFillColour is currently set to an index of a colour look-up table, *NewColour* shall be set or refer to an index of a colour look-up table.

Syntax description:

```
SetFillColour          -->   Target,
                             NewColour?
Target                 -->   GenericObjectReference
NewColour              -->   NewColourIndex | NewAbsoluteColour
NewColourIndex         -->   GenericInteger
NewAbsoluteColour      -->   GenericOctetString
```

33.5 Formal description

```
LineArt Class              -->  Visible Class,
                                BorderedBoundingBox?,
                                OriginalLineWidth?,
                                OriginalLineStyle?,
                                OriginalRefLineColour?,
                                OriginalRefFillColour?
BorderedBoundingBox        -->  BOOLEAN
OriginalLineWidth          -->  INTEGER
OriginalLineStyle          -->  INTEGER
OriginalRefLineColour      -->  Colour
OriginalRefFillColour      -->  Colour
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 118 of 253

34 Rectangle Class

Description	Defines a data structure that deals with rectangles.
Base class	LineArt
Subclasses	None
Status	Concrete class

34.1 Attributes

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

34.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	This attribute shall not be encoded for this class.
OriginalContent	Ingredient	This attribute shall not be encoded for this class.
BoxSize, Position	Visible	In addition to defining the bounding box for the Rectangle, these attributes also define the size of the Rectangle itself. The Rectangle's size shall be such that it precisely fits inside its bounding box. This means that the borders of the Rectangle occupy *LineWidth* pixels on the inside of the Rectangle's bounding box.
BorderedBoundingBox	LineArt	This attribute shall not be encoded for this class. It is always be interpreted as True. Unlike for its parent class, the BorderedBoundingBox of a Rectangle (i.e. the shape of the Rectangle) shall be drawn using LineWidth, LineStyle and RefLineColour, rather than OriginalLineWidth, OriginalLineStyle and OriginalRefLineColour.

34.1.2 Own exchanged attributes

This class defines no additional exchanged attributes.

34.1.3 Own internal attributes

This class defines no additional internal attributes.

34.2 Events

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

34.3 Internal behaviours

The internal behaviours of this class are the same as those of its base class, with identical semantics.

34.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with identical semantics.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 119 of 253

34.5 Formal description

```
Rectangle Class          --> LineArt Class
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 120 of 253

35 DynamicLineArt Class

Description	Defines means to dynamically draw vectorial graphical objects
Base class	LineArt
Subclasses	None
Status	Concrete Class

35.1 Attributes

35.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	This attribute shall not be encoded for this class.
OriginalContent	Ingredient	This attribute shall not be encoded for this class.

35.1.2 Own exchanged attributes

This class defines no additional exchanged attributes.

35.1.3 Own internal attributes

This class defines no additional internal attributes.

35.2 Events

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

35.3 Internal Behaviours

This class defines no additional internal behaviour.

35.4 Effect of MHEG-5 Actions

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

SetBoxSize	This action redraws the DynamicLineArt object with its new size and cleared of all of its previous drawings, that is filled with *OriginalRefFillColour* (in the particular case where the new size is exactly the old size, this action is similar to a Clear action). When BorderedBoundingBox is set to True, the border is redrawn.
SetPosition, BringToFront, SendToBack, PutBefore, PutBehind	After each of these actions, the DynamicLineArt widget is and cleared of all of its previous drawings, that is filled with *OriginalRefFillColour*. When BorderedBoundingBox is set to True, the border is redrawn.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 121 of 253

SetLineWidth — Setting a new value of *LineWidth* shall not modify line width of all existing graphics from the DynamicLineArt object, it shall be used to render the next graphics to be drawn.

SetLineStyle — Setting a new value of *LineStyle* shall not modify line style of all existing graphics from the DynamicLineArt object, it shall be used to render the next graphics to be drawn.

SetLineColour — Setting a new value of *RefLineColour* shall not modify the colour of all existing graphics from the DynamicLineArt object, it shall be used to render the next graphics to be drawn.

SetFillColour — Setting a new value of *RefFillColour* shall not modify colour of all existing graphics from the DynamicLineArt object, it shall be used to render the next graphics to be drawn.

GetLineWidth (*LineWidthVar*) — Returns the current *LineWidth* in the variable *LineWidthVar*.

Provisions of use:

- *LineWidthVar* shall refer to an active IntegerVariable object.
- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
GetLineWidth          -->  Target,
                           LineWidthVar
Target                -->  GenericObjectReference
LineWidthVar          -->  ObjectReference
```

GetLineStyle (*LineStyleVar*) — Returns the current *LineStyle* into the variable *LineStyleVar*.

Provisions of use:

- *LineStyleVar* shall refer to an active IntegerVariable object.
- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
GetLineStyle          -->  Target,
                           LineStyleVar
Target                -->  GenericObjectReference
LineStyleVar          -->  ObjectReference
```

GetLineColour (*LineColourVar*) — Returns the current *LineColour* in the variable *LineColourVar*

Provisions of use:

- *LineColourVar* shall refer to an active OctetStringVariable object if the RefLineColour is specified as an OctetString. *LineColourVar* shall refer to an

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 122 of 253

active IntegerVariable object if the RefLineColour is specified as an Integer.

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
GetLineColour          -->  Target,
                            LineColourVar
Target                 -->  GenericObjectReference
LineColourVar          -->  ObjectReference
```

GetFillColour (*FillColourVar*)

Returns the current *RefFillColour* in the variable *FillColourVar*

Provisions of use:

- *FillColourVar* shall refer to an active OctetStringVariable object if the RefFillColour is specified as an OctetString. *FillColourVar* shall refer to an active IntegerVariable object if the RefFillColour is specified as an Integer..
- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
GetFillColour          -->  Target,
                            FillColourVar
Target                 -->  GenericObjectReference
FillColourVar          -->  ObjectReference
```

DrawArc (*X, Y, EllipseWidth, EllipseHeight, StartAngle, ArcAngle*)

Draws an arc between *StartAngle* and *StartAngle + ArcAngle*. (arc BC in Figure 12 - Illustration of drawing parameters below).

Arc is drawn in *LineColour*.

Point *X, Y* is relative to the *Position* attribute of the object. That is to say that X=0, Y=0 corresponds to the top left corner of the bounding box. Values of X and Y outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Angles are expressed in 64ths of degrees and are in the interval [0, 23039]. *ArcAngle* shall not be 0.

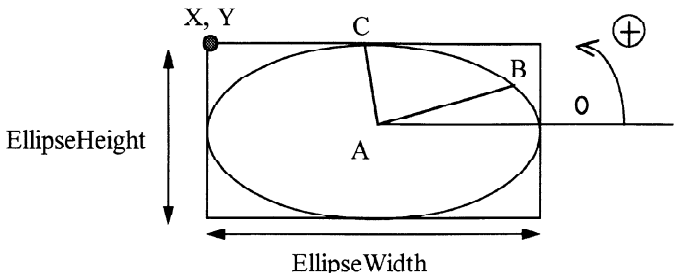


Figure 12 - Illustration of drawing parameters

Provisions of use:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 123 of 253

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawArc          -->  Target,
                      X,
                      Y,
                      EllipseWidth,
                      EllipseHeight,
                      StartAngle,
                      ArcAngle
Target           -->  GenericObjectReference
X                -->  GenericInteger
Y                -->  GenericInteger
EllipseWidth     -->  GenericInteger
EllipseHeight    -->  GenericInteger
StartAngle       -->  GenericInteger
ArcAngle         -->  GenericInteger
```

DrawSector (*X, Y, EllipseWidth, EllipseHeight, StartAngle, ArcAngle*)

Draws a sector between *StartAngle* and *StartAngle + ArcAngle*. (the surface ABC in Figure 12 - Illustration of drawing parameters above).

Lines are drawn with *RefLineColour* and the surface is filled up with *RefFillColour*.

Point *X, Y* is relative to the *Position* attribute of the object. That is to say that X=0, Y=0 corresponds to the top left corner of the bounding box. Values of X and Y outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Angles are expressed in 64ths of degrees and are in the interval [0, 23039]. *ArcAngle* shall not be 0.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawSector       -->  Target,
                      X,
                      Y,
                      EllipseWidth,
                      EllipseHeight,
                      StartAngle,
                      ArcAngle
Target           -->  GenericObjectReference
X                -->  GenericInteger
Y                -->  GenericInteger
EllipseWidth     -->  GenericInteger
EllipseHeight    -->  GenericInteger
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 124 of 253

```
StartAngle                --> GenericInteger
ArcAngle                  --> GenericInteger
```

DrawLine *(X1, Y1, X2, Y2)*

Draws a line between (*X1, Y1*) and (*X2, Y2*)

Points *X1, Y1 and X2, Y2* are relative to the *Position* attribute of the object. Values outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawLine                  --> Target,
                              X1,
                              Y1,
                              X2,
                              Y2
Target                    --> GenericObjectReference
X1                        --> GenericInteger
Y1                        --> GenericInteger
X2                        --> GenericInteger
Y2                        --> GenericInteger
```

DrawOval *(X, Y, EllipseWidth, EllipseHeight)*

Draws an ellipse bounded by the rectangle defined by the parameters (see Figure 12 - Illustration of drawing parameters above).

This ellipse is filled up with *RefFillColour*.

Point *X, Y* is relative to the *Position* attribute of the object. That is to say that X=0, Y=0 corresponds to the top left corner of the bounding box. Values of X and Y outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawOval                  --> Target,
                              X,
                              Y,
                              EllipseWidth,
                              EllipseHeight
Target                    --> GenericObjectReference
X                         --> GenericInteger
Y                         --> GenericInteger
EllipseWidth              --> GenericInteger
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 125 of 253

```
EllipseHeight              -->  GenericInteger
```

DrawPolygon (*PointList*)

Draw a closed polygon.

This polygon is filled up with *RefFillColour*.

PointList is a list of Point.

A Point is defined by X, Y coordinate, relative to the *Position* attribute of the object. Values of X and Y outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawPolygon                -->  Target,
                                PointList
Target                     -->  GenericObjectReference
PointList                  -->  Point+
Point                      -->  X,
                                Y
X                          -->  GenericInteger
Y                          -->  GenericInteger
```

DrawPolyline (*PointList*)

Draws series of joint lines.

PointList is a list of Point.

A Point is defined by X, Y coordinate, relative to the *Position* attribute of the object. Values of X and Y outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawPolyline               -->  Target,
                                PointList
Target                     -->  GenericObjectReference
PointList                  -->  Point+
Point                      -->  X,
                                Y
X                          -->  GenericInteger
Y                          -->  GenericInteger
```

DrawRectangle (*X1, Y1, X2, Y2*)

Draws a rectangle.

This rectangle is filled up with *RefFillColour*.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 126 of 253

Top Left point is (*X1, Y1*) and Bottom Right point is (*X2, Y2*). (*X1, Y1*) and (*X2, Y2*) are relative to the *Position* attribute of the object. Values outside of the bounding box are allowed, but only the part of the drawing within the bounding box shall be rendered.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
DrawRectangle          -->  Target,
                            X1,
                            Y1,
                            X2,
                            Y2
X1                     -->  GenericInteger
Y1                     -->  GenericInteger
X2                     -->  GenericInteger
Y2                     -->  GenericInteger
```

Clear Fills up the bounding box with the *OriginalRefFillColour*.

When BorderedBoundingBox is set to True, the border is not filled with *OriginalRefFillColour*.

Provisions of use:

- The *Target* object shall be an available DynamicLineArt object.

Syntax description:

```
Clear                  -->  Target
Target                 -->  GenericObjectReference
```

35.5 Formal description

```
DynamicLineArt Class    -->  LineArt Class
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 127 of 253

36 Text Class

Description	Defines attributes and behaviour of pieces of textual information.
Base class	Visible
Subclasses	EntryField, HyperText
Status	Concrete class

36.1 Attributes

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

36.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	Either this attribute or its corresponding default attribute in the Application class (TextContentHook) is mandatory for this class.
OriginalContent	Ingredient	This attribute is mandatory for this class.

36.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

OriginalFont

Indication of which font to use when initially presenting the Text object.

The OriginalFont attribute represents either a name for a font (which is resident in the MHEG-5 engine) or a reference to a Font object. In both cases, the indicated font shall be used for rendering the Text object.

When no font reference is encoded, the Text object shall be presented using the default font referenced in the active Application object, if no font is referenced there, a default font of the MHEG-5 engine shall be used.

NOTE - The OriginalFont attribute provides the initial font for a Text object. In addition, the text encoding format may contain escape sequences to switch between fonts.

- Optional attribute.
- OctetString representing a FontName, or reference to a Font object.
- Default value: Value encoded by Application.

FontAttributes

This attribute is used to set specific Font attributes such as style, character size, text colour and background colour.

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

When no FontAttributes is encoded, the Text object shall be presented using the default *FontAttributes* encoded in the active Application object, if no *FontAttributes* is referenced there, no specific attributes are set.

- Optional OctetString.
- Default value: Value encoded by Application.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 128 of 253

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

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

BackgroundColour — Indicate which colour should be used to render the background of the text object. This attribute is interpreted as a zero-based index in the colour look-up table defined by the *PaletteRef* attribute, or as a direct colour value, depending on the attribute type.

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

CharacterSet — Identification of the character set, or set of character sets, that shall be used by default for Text rendering. This Integer shall be encoded with a value representing the character set. The application domain shall define a range of CharacterSet and its semantics.

NOTE - The *CharacterSet* attribute provides the initial character set for a Text object. In addition, the text encoding format may contain escape sequences to switch between character sets.

- Optional Integer.
- Default value: The value of *CharacterSet* attribute from the Application object, if that attribute is specified.

HorizontalJustification — The *HorizontalJustification* attribute indicates how the text lines are justified relative to the vertical edges of the bounding box defined by the *BoxSize* and *Position* attributes of the Text object.

This attribute may be ignored if a coded representation of the text itself has the same functionality to specify this type of rendering. The application domain based on this part of ISO/IEC 13522 shall define each ContentHook for which the attribute be ignored.

- Optional attribute - one of *start* / *end* / *centre* / *justified.*
- Default value: *start.*

VerticalJustification — The *VerticalJustification* attribute indicates how the text lines are justified relatively to the horizontal edges of the bounding box defined by the *BoxSize* and *Position* attributes of the Text object.

This attribute may be ignored if a coded representation of the text itself has the same functionality to specify this type of rendering. The application domain based on this part of ISO/IEC 13522 shall define each ContentHook for which the attribute be ignored.

- Optional attribute - one of *start* / *end* / *centre* / *justified.*
- Default value: *start.*

LineOrientation — The *LineOrientation* attribute is combined with the *StartCorner* attribute to determine

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 129 of 253

the way characters are organised into lines, and lines into sequences of lines.

This attribute may be ignored if a coded representation of the text itself has the same functionality to specify this type of rendering. The application domain based on this part of ISO/IEC 13522 shall define each ContentHook for which the attribute be ignored.

- Optional attribute - one of *vertical* | *horizontal.*
- Default value: *horizontal.*

NOTE - For *vertical* line orientation, the character orientation is supposed to be normal (i.e., the entire line is rotated 90 degrees from *horizontal),* unless the contents of the *Text* specify otherwise.

StartCorner The *StartCorner* attribute contains an identification of the corner of the presentation area where the text rendering should start.

This attribute may be ignored if a coded representation of the text itself has the same functionality to specify this type of rendering. The application domain based on this part of ISO/IEC 13522 shall define each ContentHook for which the attribute be ignored.

- Optional attribute - one of *upper-left* | *upper-right* | *lower-left* | *lower-right.*
- Default value: *upper-left.*

TextWrapping Indicate whether the text is wrapped at the end of the line, or whether it is clipped.

If the TextWrapping attribute is set to True, the text is wrapped. If it is set to False, it is clipped.

This attribute may be ignored if a coded representation of the text itself has the same functionality to specify this type of rendering. The application domain based on this part of ISO/IEC 13522 shall define each ContentHook for which the attribute be ignored.

- Optional Boolean.
- Default value: False.

36.1.3 Own internal attributes

This class defines the following additional internal attributes:

TextData Value of the textual content data of the Text object.

If the *Content* attribute of the Text object is an IncludedContent, *TextData* is initially set to IncludedContent.

If the *Content* attribute of the Text object is a reference to an external data source, *TextData* is initially set to an OctetString representing the content of this external data source. In this case, the value of the ContentHook attribute might be used to format the value of *TextData.*

- OctetString.
- Initial value: IncludedContent or the content of a ReferencedContent.

Font Font to use when presenting the Text object.

The *Font* attribute represents either a name for a font (which is resident in the MHEG-5 engine) or a reference to a Font object. In both cases, the indicated font shall be used for rendering the Text object.

NOTE - The Font attribute provides the initial font for a Text object. In addition, the text encoding format may contain escape sequences to switch between fonts.

- Optional attribute.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 130 of 253

- OctetString representing a FontName, or reference to a Font object.
- Initial value: Value of the OriginalFont attribute.

36.2 Events

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

36.3 Internal behaviours

The internal behaviours of this class have the same semantics as for its base class.

36.4 Effect of MHEG-5 actions

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

SetData — Execute synchronously the following actions:

1. Update the value of the *TextData* internal attribute of the target Text.
2. Execute the SetData action as defined in the Presentable class.

Provisions of use and syntax description are provided in the Ingredient class.

GetTextContent (*TextContentVar*) — Set the Variable referenced by *TextContentVar* to the value of the *Content* attribute.

NOTE - If the *Content* attribute of the Text object is included, *TextContentVar* returns the text OctetString; if the *Content* attribute of the Text object is referenced, *TextContentVar* returns that object reference.

Provisions of use:

- *TextContentVar* shall refer to an active OctetStringVariable object or an active ContentRefVariable object.
- The *Target* object shall be an available Text object.

Syntax description:

```
GetTextContent          -->  Target,
                             TextContentVar
Target                  -->  GenericObjectReference
TextContentVar          -->  ObjectReference
```

GetTextData (*TextDataVar*) — Set the Variable referenced by *TextDataVar* to the value of the *TextData* attribute.

Provisions of use:

- *TextDataVar* shall refer to an active OctetString object.
- The *Target* object shall be an available Text object.

Syntax description:

```
GetTextData             -->  Target,
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 131 of 253

```
                                TextDataVar
Target                  -->     GenericObjectReference
TextDataVar             -->     ObjectReference
```

SetFontRef (*NewFont*)

Change the character font used to present a text.

Execute the following sequence of actions:

1. Set the value of the *Font* attribute to *NewFont*.
2. If the target Text object is active, redraw the target object immediately by taking into account the new value of *Font* and according to its position in the *DisplayStack* of the active Application object.

NewFont might be a reference to a Font object or a font name. This part of ISO/IEC 13522 does not define how font names are managed by the MHEG-5 engine.

Provisions of use:

- The *Target* object shall be an available Text object.
- If *NewFont* references a Font object, this Font object shall be available.

Syntax description:

```
SetFontRef              -->     Target,
                                NewFont
Target                  -->     GenericObjectReference
NewFont                 -->     NewFontName | NewFontReference
NewFontName             -->     GenericOctetString
NewFontReference        -->     GenericObjectReference
```

36.5 Formal description

```
Text Class              -->     Visible Class,
                                OriginalFont?,
                                FontAttributes?,
                                TextColour?,
                                BackgroundColour?,
                                CharacterSet?,
                                HorizontalJustification?,
                                VerticalJustification?,
                                LineOrientation?,
                                StartCorner?,
                                TextWrapping?
OriginalFont            -->     OctetString | ObjectReference
FontAttributes          -->     OctetString
TextColour              -->     Colour
BackgroundColour        -->     Colour
CharacterSet            -->     INTEGER
Horizontal-             -->     start | end | centre | justified
Justification
VerticalJustification   -->     start | end | centre | justified
LineOrientation         -->     vertical | horizontal
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 132 of 253

```
StartCorner             --> upper-left | upper-right | lower-
                            left | lower-right
TextWrapping            --> BOOLEAN
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 133 of 253

37 Stream Class

Description	Defines the behaviour of a composition of continuous media (Video, Audio and RTGraphics) that are presented in synchronisation.
Base class	Presentable
Subclasses	None
Status	Concrete class

37.1 Attributes

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

37.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	Either this attribute or its corresponding default attribute in the Application class (StreamContentHook) is mandatory for this class.
OriginalContent	Ingredient	This attribute is mandatory for this class. It contains a reference to a whole multiplex of synchronised media.

37.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

Multiplex — List of inclusion of Video, Audio and RTGraphics objects that are intended to be presented simultaneously. These are called StreamComponents below. Each stream component has a tag, which is used to identify it uniquely within the stream.

The *InitiallyActive* attribute of each StreamComponent determines whether the corresponding elementary stream is automatically played as a result of the Stream multiplex being activated for the first time.

Note that all behaviour associated with synchronisation is always accessed via the Stream class; for example, it is not possible to target the SetCounterPosition action to an Audio object.

- Sequence of inclusions of Video, Audio and RT-Graphics objects

Storage — Indicate whether the composition of continuous media is loaded into memory before rendering or if it is presented directly off the stream coming, for instance, from a server. For the MHEG-5 engine, the difference in handling is that in the *memory* case, the MHEG-5 engine shall synchronise the stream, whereas in the *stream* case, the stream is synchronised by the server.

- Optional attribute - one of *memory* / *stream*.
- Default value: *stream*.

Looping — Number of performances of the Stream object.

In the counting of loops, actual counting takes place as the Stream reaches its end. As a result SetCounterPosition action shall be interpreted in a given loop.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 134 of 253

When a Stream is played and stopped before all the loops are done, the next play action shall continue from that loop (and from that CounterPosition, unless it is set by a specific action); in other words the attribute Looping shall represent the total number of loops.

When a Stream has reached the end of all loops and is stopped, a new activation shall play the Stream as if it were activated for the first time, that is looping according to the Looping attribute.

- Optional Integer.
- Default value: 1.
- Special value: 0 means *infinity*.

37.1.3 Own internal attributes

This class defines the following additional internal attributes:

Speed Rate at which the composition of continuous media is presented.

The *Speed* attribute is a rational number, represented as two Integers *(a, b)*. Speed is defined as *a/b*. The semantics of the following values differ depending on whether the MHEG-5 engine has access to an underlying layer to support trick modes.

NOTE - Trick modes might be provided by the ISO/IEC 13818-6 DSM-CC protocol.

	No Trick Mode	*Trick Mode*
-1/1	Treated as 0.	Reverse.
		Play backwards at normal speed.
0/1	Stop.	Stop.
	Freeze at current position.	Freeze at current position.
1/1	Normal play.	Normal play.
	Play from a point that is not under the control of the MHEG-5 engine.	Play from the point where the stream was when it left its previous mode.

Other values of Speed are allowed (e.g., 1/2 or 2/1) in trick mode. In the case of no trick mode, such other value of Speed shall be treated as normal play (1/1) when they are positive and as stop (0/1) when they are negative or null.

- Pair of Integers: a Numerator and an optional Denominator that defaults to 1.
- Default value: 1/1.

CounterPosition Current temporal position of the Stream within the duration of a stream at normal speed.

This attribute is expressed in StreamCounterUnits.

The actual definition of the StreamCounterUnit is out of the scope of this part of ISO/IEC 13522.

- Integer.
- Default value: 0.

CounterEndPosition Position of the last frame of a stream played at normal speed. The stream will stop automatically when it hits this position. The *StreamStopped* event is generated.

This attribute is expressed in StreamCounterUnits.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 135 of 253

The actual definition of the StreamCounterUnit is out of the scope of this part of ISO/IEC 13522.

- Integer.
- Default value: -1, meaning EndOfStream.

CounterTriggers List of values representing the counter positions of the Stream where the stream player shall generate *CounterTrigger* events for this Stream.

Each trigger has a unique identifier within the *CounterTriggers* list and a counter position expressed in StreamCounterUnits.

- Sequence of the following data structures:
  - trigger identifier: Integer,
  - counter position: Integer.
- Initial value: Empty sequence.

37.2 Events

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

StreamEvent This event is automatically generated by the Stream player when the Stream multiplex crosses a specific marker. The marker is recognised on the basis of a tag that may be encoded within the Stream content data structure. There might be several markers with the same identity along a Stream multiplex.

- Associated data: StreamEventTag - OctetString.

NOTES

1 According to the semantics of the Link class, a Link set to trigger on StreamEvents from a specific stream object that does not have its EventData set will trigger on all StreamEvents from that object.

2 The encoding of StreamEvents may be done using the ISO/IEC 13818-6 DSM-CC protocol.

StreamPlaying This event is generated when a Stream multiplex has started playing. More specifically, it is generated simultaneously with the first piece of content data (video frame, audio sample) being presented to the user.

- No associated data.

StreamStopped This event is generated when a Stream multiplex has stopped playing. More specifically, it is generated as soon as the last piece of content data (video frame, audio sample) 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.

CounterTrigger This event is automatically generated by the Stream player when the *CounterPosition* of the Stream object crosses a value set with the SetCounterTrigger action. There might be several CounterTriggers triggered at the same counter position of a Stream.

- Associated data: TriggerIdentifier - Integer.

NOTE - The encoding of counter position within the stream may be done using the

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 136 of 253

ISO/IEC 13818-6 DSM-CC protocol.

37.3 Internal behaviours

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

Preparation Execute the following sequence of actions:

1. Apply the first three steps of the *Preparation* behaviour as defined in the Root class.
2. Apply the *Activation* behaviour to all StreamComponents of the Stream that have the InitiallyActive attribute set to True, in the order they appear in the Stream Multiplex.
3. Apply steps four to six of the *Preparation* behaviour as defined in the Root class.

Destruction Execute the following sequence of actions:

1. Apply the *Destruction* behaviour to all StreamComponents of the Stream, in the reverse order they appear in the Stream Multiplex.
2. Apply the *Destruction* behaviour as defined in the base class.

Activation Execute the following sequence of actions:

1. Apply the *Activation* behaviour as defined in the base class.
2. Start playing all active StreamComponents.
3. Set the *RunningStatus* attribute to True and generate an *IsRunning* event.

The Activation and Deactivation of StreamComponents like Audio, Video or RTGraphics corresponds to enabling or disabling them in a stream. For instance when a Stream plays, activating an Audio shall make that component audible (in addition to the others).

Deactivation Execute the following sequence of actions:

1. If the RunningStatus attribute of the object is False, ignore the behaviour, otherwise:
2. Stop playing all active StreamComponents.
3. Execute *Deactivation* behaviour as defined in the base class.

37.4 Effect of MHEG-5 actions

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

SetData, Clone — SetData and Clone shall not be targeted to Stream.

SetCounterTrigger (*TriggerIdentifier*, *NewCounterValue*) — Update the list of CounterTriggers of a Stream object.

Execute the following sequence of actions:

1. Update the set of triggers of the *CounterTriggers* list of the target Stream, according to the following rules:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 137 of 253

a) If *TriggerIdentifier* is the identifier of an existing trigger in *CounterTriggers*, the new trigger replaces the previous one.

b) If there is no trigger with identifier *TriggerIdentifier* in *CounterTriggers*, insert a new trigger with identifier *TriggerIdentifier* and value *NewCounterValue* in *CounterTriggers*.

c) If *NewCounterValue* is not encoded and there is a trigger with identifier *TriggerIdentifier* in *CounterTriggers*, remove this trigger from the *CounterTriggers* list.

d) If *NewCounterValue* is not encoded and there is no trigger with identifier *TriggerIdentifier* in *CounterTriggers*, discard this action.

2. If the target Stream object is active, the MHEG-5 engine shall generate *CounterTrigger* events according to the new value of the *CounterTriggers* list.

Provisions of use:

- The *Target* object shall be an available Stream object.

Syntax description:

```
SetCounterTrigger     -->  Target,
                           TriggerIdentifier,
                           NewCounterValue?
Target                -->  GenericObjectReference
TriggerIdentifier     -->  GenericInteger
NewCounterValue       -->  GenericInteger
```

SetSpeed (*NewSpeed*)

Change the presentation speed of a stream.

Execute the following sequence of actions:

1. Set the value of the *Speed* attribute of the target Stream object to *NewSpeed*.
2. If the target Stream is active, update immediately the rendering of the stream by taking into account the new value of the *Speed* attribute.

The *NewSpeed* attribute is defined as a ratio Numerator/Denominator.

NOTES:

1 As mentioned earlier, in a broadcast environment, play and stop shall start when possible in the broadcasted stream. In other cases, setting new speed (including normal play and stop) shall take place at the current counter position, or as close to it as possible (e.g. next I-frame). Counter position can be set by the appropriate action.

2 If *Trick Modes* are not supported by the engine, the value of *Speed* can still be set to any value, the engine will interpret them as explained in the *Speed* internal attribute.

Provisions of use:

- The *Target* object shall be an available Stream object.

Syntax description:

```
SetSpeed              -->  Target,
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 138 of 253

```
                                NewSpeed
Target              -->  GenericObjectReference
NewSpeed            -->  Rational
Rational            -->  Numerator,
                         Denominator?
Numerator           -->  GenericInteger
Denominator         -->  GenericInteger
```

SetCounterPosition (*NewCounter-Position*)

Change the current position within a stream.

Execute the following sequence of actions:

1. If the MHEG-5 engine is not provided with an underlying presentation layer that supports trick modes, discard this action.
2. Set the value of the *CounterPosition* attribute of the target Stream to *NewCounterPosition*.
3. If the target Stream is active, skip immediately to the new position without changing the *RunningStatus* of the target Stream.

NOTE - Stream events are not generated because of a SetCounterPosition action.

Provisions of use:

- The *Target* object shall be an available Stream object.
- *NewCounterPosition* shall indicate a valid position within the target Stream.

Syntax description:

```
SetCounterPosition  -->  Target,
                         NewCounterPosition
Target              -->  GenericObjectReference
NewCounterPosition  -->  GenericInteger
```

SetCounterEnd-Position (*NewCounterEnd-Position*)

Change the end position of a stream.

Execute the following sequence of actions:

1. If the MHEG-5 engine is not provided with an underlying presentation layer that supports trick modes, discard this action.
2. Set the value of the *CounterEndPosition* attribute of the target Stream to *NewCounterEndPosition*.
3. If the target Stream is active and *NewCounterEndPosition* is already passed, stop the target Stream.

Provisions of use:

- The *Target* object shall be an available Stream object.
- *NewCounterPosition* shall indicate a valid position within the target Stream.

Syntax description:

```
SetCounterPosition  -->  Target,
                         NewCounterPosition
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 139 of 253

```
Target                --> GenericObjectReference
NewCounterPosition    --> GenericInteger
```

37.5 Formal description

```
Stream Class          -->  Presentable Class,
                           Multiplex,
                           Storage?,
                           Looping?
Multiplex             -->  StreamComponent+
StreamComponent       -->  Audio class | Video class |
                           RTGraphics class
Storage               -->  memory | stream
Looping               -->  INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 140 of 253

38 Audio Class

Description	Defines the attributes and behaviour of an elementary audio stream of a Stream multiplex. The Audio object shall be a StreamComponent of a Stream object.
Base class	Presentable
Subclasses	None
Status	Concrete class

38.1 Attributes

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

38.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
RunningStatus	Root	This attribute expresses that this object is enabled or disabled to play when the Stream object containing it is active. The object is only played when this attribute is True and the RunningStatus of the Stream is True.
ContentHook, OriginalContent, Shared	Ingredient	These attributes shall not be encoded for this class.

38.1.2 Own exchanged attributes

This class defines the following additional exchanged attribute:

OriginalVolume — Volume of the Audio object when it is first available.

The OriginalVolume attribute is expressed in dB, where 0 dB is defined to be the standard volume for playing back audio. The precise accuracy of the volume rendering is outside the scope of this part of ISO/IEC 13522.

- Optional Integer.
- Default value: 0.

ComponentTag — A unique identifier for the elementary audio stream within a stream of multiplexed media.

- Integer.

38.1.3 Own internal attributes

This class defines the following additional internal attribute:

Volume — Current volume of the Audio object, defined in the same way as OriginalVolume.

- Integer.
- Initial value: *OriginalVolume.*

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 141 of 253

38.2 Events

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

38.3 Internal behaviours

This class defines no additional internal behaviours.

38.4 Effect of MHEG-5 actions

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

SetData, Clone — SetData and Clone shall not be targeted to Audio.

SetVolume (*NewVolume*) — Change the volume of an audio.

Execute the following sequence of actions:

1. Set the *Volume* attribute of the target Audio object to *NewVolume*.
2. If the elementary audio stream identified by the ComponentTag attribute is being played, update the rendering of the Audio by taking into account the new *Volume*.

Provisions of use:

- The *Target* object shall be an available Audio object.

Syntax description:

```
SetVolume       -->   Target,
                      NewVolume
Target          -->   GenericObjectReference
NewVolume       -->   GenericInteger
```

GetVolume (*VolumeVar*) — Returns the volume of an Audio.

Provisions of use:

- The *Target* object shall be an available Audio object.
- *VolumeVar* shall be an active IntegerVariable object.

Syntax description:

```
GetVolume       -->   Target,
                      VolumeVar
Target          -->   GenericObjectReference
VolumeVar       -->   ObjectReference
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 142 of 253

38.5 Formal description

```
Audio Class             -->  Presentable Class,
                             ComponentTag
                             OriginalVolume?
ComponentTag            -->  INTEGER
OriginalVolume          -->  INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 143 of 253

39 Video Class

Description	Defines the attributes and behaviour of an elementary video stream of a Stream multiplex. The Video object shall be a StreamComponent of a Stream object.
Base class	Visible
Subclasses	None
Status	Concrete class

39.1 Attributes

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

39.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
RunningStatus	Root	This attribute expresses that this object is enabled or disabled to play when the Stream object containing it is active. The object is only played when this attribute is True and the RunningStatus of the Stream is True.
ContentHook, OriginalContent, Shared	Ingredient	These attributes shall not be encoded for this class.
OriginalPaletteRef	Visible	This attribute shall not be encoded for this class.

39.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

Termination	This attribute indicates whether the last frame of the video shall disappear when the presentation of the video finishes, or whether it shall freeze. • Optional attribute - One of *freeze* I *disappear.* • Default value: *disappear.*
ComponentTag	A unique identifier for the elementary video stream within a stream of multiplexed media. • Integer.

39.1.3 Own internal attributes

This class defines no additional internal attribute.

39.2 Events

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

39.3 Internal behaviours

This class defines no additional internal behaviours.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 144 of 253

39.4 Effect of MHEG-5 actions

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

SetData, Clone — SetData and Clone shall not be targeted to Video.

ScaleVideo (*XScale*, *YScale*) — If the MHEG-5 engine implements the Scaling option, the effect of this action is to adapt the rendering of the Video so that it fits to the *XScale* and *YScale* dimensions.

XScale and YScale parameters represent the final dimensions of the Video in pixel numbers. Thus, the graphical representation of the Video may not keep its original aspect ratio.

Note that this action does not affect the *BoxSize* internal attribute of the Video object.

Provisions of use:

- The *Target* object shall be an available Video object.
- *XScale* and *YScale* shall be positive Integers.

Syntax description:

```
ScaleVideo          -->  Target,
                         XScale, YScale
Target              -->  GenericObjectReference
XScale, YScale      -->  GenericInteger
```

39.5 Formal description

```
Video Class             -->  Visible Class,
                             ComponentTag
                             Termination?
Termination             -->  freeze | disappear
ComponentTag            -->  INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 145 of 253

40 RTGraphics Class

Description	Defines the attributes and behaviour of non-persistent graphics objects, defined as of an elementary stream of a Stream multiplex. The RTGraphics object shall be a StreamComponent of a Stream object.
Base class	Visible
Subclasses	None
Status	Concrete class

40.1 Attributes

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

40.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
RunningStatus	Root	This attribute expresses that this object is enabled or disabled to play when the Stream object containing it is active. The object is only played when this attribute is True and the RunningStatus of the Stream is True.
ContentHook, OriginalContent, Shared	Ingredient	These attributes shall not be encoded for this class.

40.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

ComponentTag — A unique identifier for the elementary RTGraphics stream within a stream of multiplexed media.

- Integer.

Termination — This attribute indicates whether the last image of the graphics shall disappear when the presentation finishes, or whether it shall freeze.

- Optional attribute - One of *freeze* | *disappear.*
- Default value: *disappear.*

40.1.3 Own internal attributes

This class defines no additional internal attribute.

40.2 Events

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

40.3 Internal behaviours

This class defines no additional internal behaviours.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 146 of 253

40.4 Effect of MHEG-5 actions

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

SetData, Clone	SetData and Clone shall not be targeted to RTGraphics.

40.5 Formal description

```
RTGraphics Class        -->  Visible Class,
                             ComponentTag
                             Termination?
ComponentTag            -->  INTEGER
Termination             -->  freeze | disappear
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 147 of 253

41 Interactible Class

Description	Defines functionality associated with an interaction behaviour of Visibles.
Base class	None (mix-in class)
Subclasses	Slider, HyperText, EntryField, Button
Status	Abstract class

41.1 Attributes

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

41.1.1 Inherited attributes

This class has no inherited attributes.

41.1.2 Own exchanged attributes

This class defines the following exchanged attributes:

EngineResp

Determine where the responsibility lies for generating visual feedback to the user as the result of changes to the internal attribute *HighlightStatus*.

If *EngineResp* is set to True, the MHEG-5 engine shall generate visual feedback to the user when there is a state change in the internal attribute *HighlightStatus*. When *EngineResp* is set to False, the engine shall generate no such visual feedback.

The exact nature of the visual feedback is outside the scope of this part of ISO/IEC 13522.

- Optional Boolean.
- Default value: True.

HighlightRefColour

Reference colour for visual feedback generated when the *HighlightStatus* attribute is True. The actual colour used for rendering the highlight is outside the scope of this part of ISO/IEC 13522; however, it is recommended that an MHEG-5 engine attempts to render the highlight using *HighlightRefColour*.

If the OriginalPaletteRef attribute is encoded, the HighlightRefColour attribute shall be a zero-based index of a colour in the colour look-up table defined by that Palette. Otherwise, it shall be an OctetString encoding an actual colour.

- Optional Integer or OctetString.
- Default value: Value encoded by Application.

41.1.3 Own internal attributes

This class defines the following internal attributes:

HighlightStatus

This attribute is associated with a certain type of visual feedback to the user.

When both *HighlightStatus* and *EngineResp* are True, the MHEG-5 engine shall generate a visual feedback to the user, for instance in the form of a line drawn around the Interactible object. In all other cases, no such feedback shall be generated. For the generation of this visual feedback, the HighlightRefColour may be

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 148 of 253

used.

NOTES

1 Although this visual feedback itself does not change the behaviour of the object, this attribute may be used to signal to the user that the Interactible is ready to be interacted with, for instance in an implementation of jumping-highlight navigation.

2 The only way to change the *HighlightStatus* attribute is through the SetHighlightStatus action.

- Boolean value.
- Initial value: False.

InteractionStatus This attribute describes whether or not the Interactible is currently being interacted with.

If *InteractionStatus* is False, the Interactible is not currently being interacted with by the user. Handling of user input events takes place normally. See clause 53.

If *InteractionStatus* is True, the Interactible is currently being interacted with by the user. As a consequence, no events of the type UserInput will be generated by the active Scene object. These events shall be handled directly by the Interactible.

At any time, at most one Interactible shall have its *InteractionStatus* set to True.

NOTES

1 The only way to change the *InteractionStatus* internal attribute is by using the SetInteractionStatus action.

2 Although Links that trigger on UserInput events cannot fire when the *InteractionStatus* internal attribute is True, other Links may still fire. This makes it possible, for instance, to implement time-outs for the interaction process.

- Boolean value.
- Initial value: False.

41.2 Events

This class defines the following events:

Interaction-Completed This event is generated as a result of a change in an Interactible. The event is generated only once per interaction, i.e., it is generated only when the *InteractionStatus* internal attribute returns to False after an interaction.

- No associated data.

HighlightOn This event is generated when the *HighlightStatus* attribute of the Interactible changes from False to True.

- No associated data.

HighlightOff This event is generated when the *HighlightStatus* attribute of the Interactible changes from True to False.

- No associated data.

CursorEnter This event is generated automatically by the MHEG-5 engine if, and only if:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 149 of 253

1. The engine implements the Free Moving Cursor option.
2. The moving cursor has entered the area defined by the Interactible.
3. The Interactible is active.

NOTE - See subclause 53.6 for more details on the generation of *CursorLeave* and *CursorEnter* events.

CursorLeave This event is generated automatically by the MHEG-5 engine if, and only if:

1. The engine implements the Free Moving Cursor option.
2. The moving cursor has left the area defined by the Interactible.
3. The Interactible is active.

NOTE - See subclause 53.6 on main mechanisms for more details on the generation of *CursorLeave* and *CursorEnter* events.

41.3 Internal behaviours

This class defines the following internal behaviour:

Interaction Execute the following steps synchronously:

1. Set the *InteractionStatus* internal attribute to True.
2. Generate visual feedback to the user that signals the fact that the Interactible may now be interacted with.

NOTE - Most of the Interactible subclasses expand on this behaviour.

41.4 Effect of MHEG-5 actions

This class defines the following applicable MHEG-5 actions:

SetInteractionStatus (*NewInteraction-Status*) This action is used to influence the *InteractionStatus* internal attribute.

Execute the following sequence of actions:

1. If *NewInteractionStatus* is set to True,
   a) if the target Interactible or any other Interactible in the current Scene has its *InteractionStatus* attribute set to True, discard the action. Otherwise,
   b) apply the *Interaction* behaviour of the target Interactible.
2. If *NewInteractionStatus* is set to False,
   a) if the target Interactible has its *InteractionStatus* set to False, discard the action. Otherwise,
   b) immediately interrupt the *Interaction* behaviour that is taking place for the target Interactible. The state of the target Interactible after the *Interaction* behaviour is interrupted shall reflect any interaction which has taken place up to the point where the interaction was interrupted.

Provisions of use:

- The *Target* object shall be an active Interactible object.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 150 of 253

Syntax description:

```
SetInteractionStatus    --> Target,
                            NewInteractionStatus
Target                    > GenericObjectReference
NewInteractionStatus    --> GenericBoolean
```

GetInteractionStatus (*InteractionStatus-Var*)

Set the Variable referenced by *InteractionStatusVar* to the value of the *InteractionStatus* attribute.

Provisions of use:

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

Syntax description:

```
GetInteractionStatus      -->   Target,
                                InteractionStatusVar
Target                    -->   GenericObjectReference
InteractionStatusVar      -->   ObjectReference
```

SetHighlightStatus (*NewHighlight-Status*)

Change the highlighted state of an Interactible.

Execute synchronously the following sequence of actions:

1. If the current *HighlightStatus* is equal to *NewHighlightStatus*, discard the action. Otherwise,
2. Set the *HighlightStatus* attribute of the target Interactible to *NewHighlightStatus*.
3. If the target Interactible is active, if EngineResp is set to True and if *HighlightStatus* is set to True, redraw the target Interactible to present the visual feedback associated to the highlighted state.
4. If the target Interactible is active, if EngineResp is set to True and if *HighlightStatus* is set to False, redraw the target Interactible to remove the visual feedback associated to the highlighted state.

Provisions of use:

- The *Target* object shall be an available Interactible object.

Syntax description:

```
SetHighlightStatus      --> Target,
                            NewHighlightStatus
Target                  --> GenericObjectReference
NewHighlightStatus      --> GenericBoolean
```

GetHighlightStatus (*HighlightStatusVar*)

Set the Variable referenced by *HighlightStatusVar* to the value of the *HighlightStatus* attribute.

Provisions of use:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 151 of 253

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

Syntax description:

```
GetHighlightStatus      -->  Target,
                             HighlightStatusVar
Target                  -->  GenericObjectReference
HighlightStatusVar      -->  ObjectReference
```

41.5 Formal description

```
Interactible Class      -->  EngineResp?,
                             HighlightRefColour?,
EngineResp              -->  BOOLEAN
HighlightRefColour      -->  Colour
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 152 of 253

42 Slider Class

Description	Defines an interaction widget used to set a position within a linear range.
Base classes	Visible, Interactible
Subclasses	None
Status	Concrete class

42.1 Attributes

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

42.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	This attribute shall not be encoded for this class.
OriginalContent	Ingredient	This attribute shall not be encoded for this class.
BoxSize, Position	Visible	In addition to defining the bounding box of the Slider, these attributes also define the actual size of the Slider. This size shall be such that the Slider completely fills its bounding box.

42.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

Orientation — Orientation of the main axis of the Slider.

The orientation specifies the direction in which the Slider moves from the minimum towards the maximum value.

Although the exact rendering of the Slider object is not specified in detail by this part of ISO/IEC 13522, it shall be rendered such that its orientation conforms to this attribute.

- Possible values: *left* / *right* / *up* / *down*

InitialValue — The initial value of the Slider, i.e., the value it has when it is first activated after it has been prepared.

The value shall be an Integer and it shall be consistent with MinValue, MaxValue, and StepSize in the manner described below.

- Optional Integer.
- Default value: MinValue.

MinValue — Lowest value that the *SliderValue* attribute may be set to.

The value shall be consistent with InitialValue, MaxValue, and StepSize in the manner described below.

- Optional Integer.
- Default value: 1.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 153 of 253

MaxValue

Greatest value that the *SliderValue* attribute may be set to.

The value shall be consistent with InitialValue, MinValue, and StepSize in the manner described below.

- Integer value.
- Default value: None.

InitialPortion

Represents a portion of the range [MinValue, MaxValue]. This value shall be smaller than or equal to the number (MaxValue – MinValue).

This value shall be encoded if and only if the value of the SliderStyle attribute is *proportional, in which case the following double inequality is always True:*

$$MinValue \leq InitialValue \leq MaxValue - InitialPortion$$

- Optional Integer.
- Default value: None.

StepSize

The smallest value by which the value of the *SliderValue* internal attribute may be increased or decreased.

The value shall be a positive Integer, and it shall be consistent with InitialValue, MinValue, and MaxValue in the following manner:

All values that the slider may take are expressed as

vi = MinValue + i x StepSize,

where i = 0 ... (MaxValue - MinValue)/StepSize.

The attributes InitialValue, MinValue, MaxValue and StepSize shall conform to the following criteria:

InitialValue = MinValue + M x StepSize, for some M – 0 ... (MaxValue - MinValue)/StepSize.

MinValue < MaxValue

N x StepSize = (MaxValue - MinValue), where N is some positive Integer.

- Optional Integer.
- Default value: 1.

SliderStyle

This attribute may take the value *normal*, *thermometer*, and *proportional*. The *SliderStyle* attribute influences the rendering of the Slider in the following way:

- If the *SliderStyle* is set to *normal*, the Slider is rendered as a «marker,» which is positioned on a «main axis» at the position corresponding to the *SliderValue* attribute.
- If the *SliderStyle* is set to *thermometer*, the Slider is rendered as a «main axis,» which is filled from its beginning to the position corresponding to the *SliderValue* attribute.
- If the *SliderStyle* is set to *proportional*, the Slider is rendered as a «main axis,» which is filled from the position corresponding to the *SliderValue* attribute to the position corresponding to the sum of the *SliderValue* and the *Portion* internal attributes.

This part of ISO/IEC 13522 does not specify exactly what this rendering should look like. The following picture is provided by way of example only:

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 154 of 253



Figure 13: Examples of Sliders in different *SliderStyles*

- Optional attribute.
- Possible values: *normal* / *thermometer* / *proportional.*
- *Default value: normal.*

SliderRefColour — Specify a colour that may be used by the engine to render the Slider object.

The *SliderRefColour* value is expressed either as an absolute colour value or as a zero-based index in a colour look-up table. In the latter case, the Slider object must have the *PaletteRef* attribute encoded, which is then used to translate the index to an actual colour value.

Exactly how this colour is used to render the Slider is not specified by this part of ISO/IEC 13522. It is provided as a hint to the MHEG-5 engine on the colour scheme to use when rendering the Slider.

The actual colour resolution in the rendering process is outside the scope of this part of ISO/IEC 13522.

- Optional attribute.
- Default value: Value encoded by Application.

42.1.3 Own internal attributes

This class defines the following additional internal attributes:

SliderValue — The current value of the slider.

Portion — The current value of the attribute that governs the rendering of the Slider when the *SliderStyle* is *proportional.*

If the *SliderStyle* is set to *proportional*, the Slider is rendered as a «main axis» which is filled from the position corresponding to the *SliderValue* attribute to the position corresponding to the sum of the *SliderValue* and the *Portion* internal attributes.

42.2 Events

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

42.3 Internal behaviour

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

Interaction — Execute the following sequence of actions:

1. Apply the Interaction behaviour as defined in the Interactible class.
2. Allow the user to interact with the Slider object by moving the marker along the main axis. Exactly how this user interaction takes place is not specified

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 155 of 253

by this part of ISO/IEC 13522. However, the smallest marker displacement shall be proportional to the value of the *StepSize* attribute.

3. When the marker has stopped at a new position,
   a) set the *SliderValue* attribute to a value that corresponds to the new marker position,
   b) set the *InteractionStatus* attribute to False, and
   c) generate an *InteractionCompleted* event.

42.4 Effect of MHEG-5 actions

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

Step (*NbOfSteps*)

Set a new value to a slider relatively to its current value.

Execute the following sequence of actions:

1. If *NbOfSteps* is positive, increase the value of *SliderValue* by *NbOfSteps* X *StepSize*.
2. If *NbOfSteps* is negative, decrease the value of *SliderValue* by *NbOfSteps* X *StepSize*.
3. If the target Slider is active, redraw the Slider using the new value of *SliderValue* and according to its position in the *DisplayStack* of the active Application object.
4. Generate an *InteractionCompleted* event.

Provisions of use:

- The *Target* object shall be an available Slider object.
- *NbOfSteps* shall be set so that:

  MinValue <= (*NbOfSteps* X *StepSize*) + *SliderValue*, and

  (*NbOfSteps* X *StepSize*) + *SliderValue* + *Portion* <= MaxValue

  The value of *Portion* in the expressions above shall be taken to be 0 if the SliderStyle is not proportional.

Syntax description:

```
Step                --> Target,
                         NbOfSteps
Target              --> GenericObjectReference
NbOfSteps           --> GenericInteger
```

SetSliderValue (*NewSliderValue*)

Set an absolute value to a slider.

Execute the following sequence of actions:

1. Set the *SliderValue* attribute of the target Slider to *NewSliderValue*.
2. If the target Slider is active, redraw the Slider taking into account the new value of *SliderValue* and according to its position in the *DisplayStack* of the active Application object.
3. Generate an *InteractionCompleted* event.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 156 of 253

Provisions of use:

- The *Target* object shall be an available Slider object.

  NewSliderValue shall be within the range [MinValue, MaxValue-*Portion*].

  The value of *Portion* in the expression above shall be taken to be 0 if the SliderStyle is not proportional.
- (*NewSliderValue* - MinValue) MOD StepSize shall be equal to 0.

Syntax description:

```
SetSliderValue      -->  Target,
                         NewSliderValue
Target              -->  GenericObjectReference
NewSliderValue      -->  GenericInteger
```

GetSliderValue (*SliderValueVar*)

Set the Variable referenced by *SliderValueVar* to the value of the *SliderValue* attribute.

Provisions of use:

- The *Target* object shall be an available Slider object.
- *SliderValueVar* shall refer to an active IntegerVariable object.

Syntax description:

```
GetSliderValue      -->  Target,
                         SliderValueVar
Target              -->  GenericObjectReference
SliderValueVar      -->  ObjectReference
```

SetPortion (*NewPortion*)

Set the size of the portion represented by a slider of style *proportional*.

Execute the following sequence of actions:

1. Set the *Portion* attribute of the target Slider to *NewPortion*.
2. If the target Slider is active, redraw the Slider according to the new value of *Portion* and according to its position in the *DisplayStack* of the active Application object.
3. Generate an *InteractionCompleted* event.

Provisions of use:

- The Target object shall be an available Slider object of style *proportional*.
- *NewPortion* shall be smaller than or equal to (MaxValue - SliderValue)

Syntax description:

```
SetPortion          -->  Target,
                         NewPortion
Target              -->  GenericObjectReference
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 157 of 253

```
NewPortion          -->  GenericInteger
```

GetPortion (*PortionVar*) Set the Variable referenced by *PortionVar* to the value of the *Portion* attribute.

Provisions of use:

- The *Target* object shall be an available Slider object of style *proportional*.
- *PortionVar* shall refer to an active IntegerVariable object.

Syntax description:

```
GetPortion          -->  Target,
                         PortionVar
Target              -->  GenericObjectReference
PortionVar          -->  ObjectReference
```

42.5 Formal description

```
Slider Class        -->  Visible Class,
                         Interactible Class,
                         Orientation,
                         Max Value,
                         Min Value?,
                         InitialValue?,
                         InitialPortion?,
                         StepSize?,
                         SliderStyle?,
                         SliderRefColour?
Orientation         -->  left | right | up | down
InitialValue          >  INTEGER
InitialPortion      -->  INTEGER
Min Value           -->  INTEGER
Max Value           -->  INTEGER
StepSize            -->  INTEGER
SliderStyle         -->  normal | thermometer | proportional
SliderRefColour     -->  Colour
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 158 of 253

43 EntryField Class

Description	Defines an interaction widget used by the final user to edit and/or modify a text.
Base classes	Text, Interactible
Subclasses	None
Status	Concrete class

43.1 Attributes

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

43.1.1 Inherited attributes

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

43.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

InputType — Type of allowed characters.

If this attribute is set to *alpha*, only characters that are not digits (i.e., 0123456789) shall be accepted as entries. If this attribute is set to *numeric*, only characters that are digits shall be accepted as be entries. If this attribute is set to *any*, all characters shall be accepted. If this attribute is set to *listed*, only such characters shall be accepted that are provided in the CharList attribute of the EntryField. This provides a possibility to customise the input filter.

NOTE - The value *any* also allows for entry of symbols (e.g., &*$#@!).

- Possible values: *alpha* / *numeric* / *any* / *listed*.
- *Default value: any.*

CharList — Characters that might be entered in this EntryField.

This attribute shall always be encoded when InputType is set to *listed*. Otherwise, it shall not be encoded.

- Optional OctetString.
- Default value: None.

ObscuredInput — Indicate how to echo back input characters. This may be used for password input.

If this attribute is set to True, the entered characters shall be echoed back to the screen in an unreadable form. Otherwise, the entered characters shall be echoed back to the screen in a readable form.

- Optional Boolean value.
- Default value: False.

MaxLength — Provide the maximum number of expected input characters. When this maximum number of characters is reached in the EntryField, an EntryFieldFull event is generated.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 159 of 253

If *MaxLength* is set to 0, the number of expected characters is undefined. In this case, the MHEG-5 engine shall provide the user with some means of terminating the interaction.

- Optional Integer.
- Default value: 0.

43.1.3 Own internal attributes

This class defines the following additional internal attributes:

EntryPoint Defines (as a zero-based index) where the next character to be entered in the EntryField shall be placed.

- If this attribute is equal to 0, the next character shall be placed before the first character of the EntryField.
- If this attribute is equal to or greater than the length of the text currently in the EntryField, the next character shall be attached to the end of the EntryField.
- For all other values n of this attribute, the next character shall be inserted after the n^{th} character of the EntryField.

- Integer value.
- Initial value: 0.

OverwriteMode Determine whether new input characters overwrite characters in the existing text or are inserted between them.

If this attribute is True, each character entered in the EntryField replaces (overwrites) the character that was previously located at the *EntryPoint*, all other characters are untouched.

If this attribute is False, the entry is inserted just before the character located at the *EntryPoint*.

If the *EntryPoint* is set to a value equal to or greater than the length of the text of the EntryField, the characters that are entered are attached to the end of the EntryField regardless of the value of this attribute.

- Boolean.
- Default value: False.

43.2 Events

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

InteractionCompleted This event is generated as a result of the ending the modification process of an EntryField.

EntryFieldFull This event is generated when an EntryField capacity is reached. This event shall only be generated when MaxLength attribute is encoded, in which case it is generated when the number of characters in the EntryField reaches MaxLength.

The event is asynchronous.

A SetData action shall not generate EntryFieldFull events.

- No associated data.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 160 of 253

43.3 Internal behaviours

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

Interaction Execute the following sequence of actions:

1. Apply the *Interaction* behaviour as defined in the Interactible class.
2. Allow the user to modify the *TextData* attribute of the EntryField object by entering characters.
3. After each character is entered, update the *TextData* attribute accordingly (taking into account the value of the *EntryPoint* and *Overwrite* attributes), update the *EntryPoint* attribute.
4. When the entry is complete (because the user terminates the entry or because the application terminates it using the SetInteractionStatus action),
   a) set the *InteractionStatus* attribute to False,
   b) generate an *InteractionCompleted* event.

43.4 Effect of MHEG-5 actions

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

SetOverwriteMode (*NewOverwrite-Mode*) Change an entry field to toggle to and from overwriting mode.

Execute the following sequence of actions:

1. If current *OverwriteMode* is equal to *NewOverwriteMode*, discard the action. Otherwise,
2. Set *OverwriteMode* of the target EntryField to *NewOverwriteMode*.
3. If the target EntryField is active, update the visible representation of the EntryField according to the new value of *OverwriteMode*.

Provisions of use:

- The *Target* object shall be an available EntryField object.

Syntax description:

```
SetOverwriteMode      -->  Target,
                           NewOverwriteMode
Target                -->  GenericObjectReference
NewOverwriteMode      -->  GenericBoolean
```

GetOverwriteMode (*OverwriteModeVar*) Set the Variable referenced by *OverwriteModeVar* to the value of the *OverwriteMode* attribute.

Provisions of use:

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

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 161 of 253

Syntax description:

```
GetOverwriteMode      -->  Target,
                           OverwriteModeVar
Target                -->  GenericObjectReference
OverwriteModeVar      -->  ObjectReference
```

SetEntryPoint (*NewEntryPoint*)

Change the position of the entry point of an entry field.

Execute the following sequence of actions:

1. Set *EntryPoint* of the target EntryField to *NewEntryPoint*.
2. If the target EntryField is active, update the visible representation of the EntryField according to the new value of *EntryPoint*.

Provisions of use:

- The *Target* object shall be an available EntryField.
- *NewEntryPoint* shall be greater than or equal to 0.

Syntax description:

```
SetEntryPoint         -->  Target,
                           NewEntryPoint
Target                -->  GenericObjectReference
NewEntryPoint         -->  GenericInteger
```

GetEntryPoint (*EntryPointVar*)

Set the Variable referenced by *EntryPoinVar* to the value of the EntryPoint attribute.

Provisions of use:

- The *Target* object shall be an available EntryField object.
- *EntryPointVar* shall refer to an active IntegerVariable object.

Syntax description:

```
GetEntryPoint         -->  Target,
                           EntryPointVar
Target                -->  GenericObjectReference
EntryPointVar         -->  ObjectReference
```

43.5 Formal description

```
EntryField Class      -->  Text Class,
                           Interactible Class,
                           InputType?,
                           CharList?,
                           ObscuredInput?,
                           MaxLength?
InputType             -->  alpha | numeric | any | listed
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 162 of 253

```
CharList            -->  OctetString
ObscuredInput       -->  BOOLEAN
MaxLength           -->  INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 163 of 253

44 HyperText Class

Description: The HyperText class is a subclass of the Text class, with the special property of allowing for portions of the text to be associated with a piece of information. These portions of text are called *anchors* in the rest of this clause.

Base classes: Text, Interactible

Subclasses: None

Status: Concrete class

44.1 Attributes

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

44.1.1 Inherited attributes

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

44.1.2 Own exchanged attributes

This class defines no additional exchanged attribute.

44.1.3 Own internal attributes

This class defines the following additional internal attributes:

LastAnchorFired Tag of the last anchor fired.

- OctetString.
- Initial value: Empty string.

44.2 Events

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

AnchorFired Signals that the user has selected one of the anchors in the HyperText object.

- Associated data: the tag of the anchor that fired - OctetString.

44.3 Internal behaviours

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

Interaction Execute the following sequence of actions:

1. Apply the *Interaction* behaviour as inherited from the Interactible class.
2. Allow the user to move the focus through the set of anchors in the HyperText object and to select the focused anchor. Each time an anchor is selected, an *AnchorFired* event is generated.
3. When the interaction is complete (either because the user terminates the

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 164 of 253

interaction or because the application terminates it using the SetInteractionStatus action),

a) set the *InteractionStatus* attribute to False,

b) generate an *InteractionCompleted* event.

44.4 Effect of MHEG-5 actions

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

GetLastAnchorFired (*LastAnchorFiredVar*)

Set the Variable referenced by *LastAnchorFiredVar* to the value of the *LastAnchorFired* attribute.

Provisions of use:

- The *Target* object shall be an available Hypertext object.
- *LastAnchorFired* shall refer to an active OctetStringVariable object.

Syntax description:

```
GetLastAnchorFired    -->  Target,
                           LastAnchorFiredVar
Target                 >   GenericObjectReference
LastAnchorFiredVar    -->  ObjectReference
```

44.5 Formal description

```
HyperText Class          -->  Text Class,
                              Interactible Class
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 165 of 253

45 Button Class

Description	Defines functionality associated with the rendering and interaction with one-state and two-state buttons.
Base class	Visible, Interactible
Subclasses	Hotspot, PushButton
Status	Abstract class

45.1 Attributes

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

45.1.1 Inherited attributes

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

Attribute Name	Defined in	Constraints and Requirements
ContentHook	Ingredient	This attribute shall not be encoded for this class.
OriginalContent	Ingredient	This attribute shall not be encoded for this class.
InteractionStatus	Interactible	This internal attribute is not defined for the Button class.
BoxSize, Position	Visible	In addition to defining the bounding box of the Button, these attributes also define the actual size of the Button. This size shall be such that the Button completely fills its bounding box.

45.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

ButtonRefColour — Specify a colour that may be used by the engine to render the Button object.

The *ButtonRefColour* attribute value is expressed either as an absolute colour value or as a zero-based index in a colour look-up table. In the latter case, the Button object must have the *PaletteRef* attribute encoded, which is then used to translate the index to an actual colour value.

Exactly how this colour is used to render the Button is not specified by this part of ISO/IEC 13522. It is provided as a hint to the MHEG-5 engine on the colour scheme to use when rendering the Button.

The actual colour resolution in the rendering process is outside the scope of this part of ISO/IEC 13522.

- Optional attribute.
- Default value: Value encoded by Application.

45.1.3 Own internal attributes

This class defines the following additional internal attributes:

SelectionStatus — Each button may store one bit of information. This attribute is True when the Button object is in the selected state. The selected state is entered as a result of invoking the Select action.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 166 of 253

- Boolean value.
- Initial value: False

45.2 Events

IsSelected This event is generated when the *SelectionStatus* of the Button changes from False to True.

- No associated data.

IsDeselected This event is generated when the *SelectionStatus* of the Button changes from True to False.

- No associated data.

45.3 Internal behaviours

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

Interaction This behaviour is not defined for the Button class.

NOTE - As a result, the Button class shall not generate any InteractionCompleted event.

Selection Execute the following sequence of actions:

1. Set the *SelectionStatus* to True.
2. If the *EngineResp* attribute is True and the Button is active, redraw the Button object taking into account the new value of SelectionStatus and according to its position in the *DisplayStack* of the active Application object.
3. Generate an *IsSelected* event.

Deselection Execute the following sequence of actions:

1. Set the *SelectionStatus* to False.
2. If the *EngineResp* attribute is True and the Button is active, redraw the Button object taking into account the new value of SelectionStatus and according to its position in the *DisplayStack* of the active Application object.
3. Generate an *IsDeselected* event.

45.4 Effect of MHEG-5 actions

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

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

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

Select Execute the following sequence of actions:

1. If *SelectionStatus* is currently set to True, disregard this action.
2. If *SelectionStatus* is currently set to False, apply the *Selection* behaviour of the

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 167 of 253

target Button.

Provisions of use:

- The *Target* object shall be an available Button.

Syntax description:

```
Select              -->  Target
Target              -->  GenericObjectReference
```

Deselect Execute the following sequence of actions:

1. If *SelectionStatus* is currently set to False, disregard this action.
2. If *SelectionStatus* is currently set to True, apply the *Deselection* behaviour of the target Button.

Provisions of use:

- The *Target* object shall be an available Button.

Syntax description:

```
Deselect            -->  Target
Target              -->  GenericObjectReference
```

45.5 Formal description

```
Button Class            -->  Visible Class,
                             Interactible Class,
                             ButtonRefColour?
ButtonRefColour         -->  Colour
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 168 of 253

46 Hotspot Class

Description	Defines invisible unlabelled rectangular areas on the screen that may interact with the user to produce *IsSelected* events.
Base class	Button
Subclasses	None
Status	Concrete class

46.1 Attributes

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

46.1.1 Inherited attributes

This class has all the attributes of its base classes, with identical semantics, except for the following attribute:

SelectionStatus 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 ButtonColour 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.

46.1.2 Own exchanged attributes

This class defines no additional exchanged attribute.

46.1.3 Own internal attributes

This class defines no additional internal attribute.

46.2 Events

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

46.3 Internal behaviours

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

46.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class. The following MHEG-5 action semantics have changed from this object's base class:

Select Execute the following sequence of actions:

1. Apply the *Selection* behaviour as defined in the base class.
2. Apply the *Deselection* behaviour as defined in the base class.

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

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 169 of 253

46.5 Formal description

```
Hotspot Class          --> Button Class
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 170 of 253

47 PushButton Class

Description	Defines labelled, largely rectangular areas on the screen that may interact with the user to produce *IsSelected* events.
Base class	Button
Subclasses	SwitchButton
Status	Concrete class

47.1 Attributes

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

47.1.1 Inherited attributes

This class has all the attributes of its base classes, with identical semantics, except for the following attribute:

SelectionStatus 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.

47.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

OriginalLabel One-line piece of text that represents the initial label of the PushButton.

- Optional OctetString.
- Default value: Empty string.

CharacterSet Identification of the character set, or set of character sets, that shall be used by default for rendering of the label. This Integer shall be encoded with a value representing the character set. The application domain shall define a range of CharacterSet and its semantics.

NOTE - The *CharacterSet* attribute provides the initial character set for a label. In addition, the text encoding format may contain escape sequences to switch between character sets.

- Optional Integer.
- Default value: The value of *CharacterSet* attribute from the Application object, if that attribute is specified.

47.1.3 Own internal attributes

This class defines the following additional internal attributes:

Label Label of the PushButton.

- Optional OctetString.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 171 of 253

- Initial value: Value of the OriginalLabel attribute.

47.2 Events

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

47.3 Internal behaviours

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

47.4 Effect of MHEG-5 actions

This class has the same set of MHEG-5 actions as its base class, with the following changes and extensions:

Select — Execute the following sequence of actions:

1. Apply the *Selection* behaviour as defined in the base class.
2. Apply the *Deselection* behaviour as defined in the base class.

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

SetLabel (*NewLabel*) — Change the label of a PushButton.

Execute the following sequence of actions:

1. Set *Label* attribute of the target PushButton to *NewLabel*.
2. If the target PushButton is active, redraw the PushButton taking into account the new value of *Label* and according to its position in the *DisplayStack* of the active Application object.

Provisions of use:

- The *Target* object shall be an available PushButton.

Syntax definition:

```
SetLabel        -->  Target,
                     NewLabel
Target          -->  GenericObjectReference
NewLabel        -->  GenericOctetString
```

GetLabel (*LabelVar*) — Set the Variable referenced by *LabelVar* to the value of the *Label* attribute.

Provisions of use:

- The *Target* object shall be an available PushButton.
- *LabelVar* shall refer to an active OctetStringVariable object.

Syntax description:

```
GetLabel        -->  Target,
                     LabelVar
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 172 of 253

```
Target                    -->  GenericObjectReference
LabelVar                  -->  ObjectReference
```

47.5 Formal description

```
PushButton Class          -->  Button Class,
                               OriginalLabel?,
                               CharacterSet?
OriginalLabel             -->  OctetString
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 173 of 253

48 SwitchButton Class

Description	Defines a data structure to deal with labelled, largely rectangular areas on the screen that may interact with the user to produce *IsSelected* and *IsDeselected* events.
Base class	PushButton
Subclasses	None
Status	Concrete class

48.1 Attributes

This class has all the attributes of its base classes, with identical semantics, except for the following attribute:

SelectionStatus 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.

48.1.1 Inherited attributes

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

48.1.2 Own exchanged attributes

This class defines the following additional exchanged attributes:

ButtonStyle Presentation style of the SwitchButton.

- Possible values: *pushbutton* | *radiobutton* | *checkbox*

48.1.3 Own internal attributes

This class defines no additional internal attributes.

48.2 Events

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

48.3 Internal behaviours

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

48.4 Effect of MHEG-5 actions

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

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 174 of 253

GetSelectionStatus (*SelectionStatusVar*)

Return the value of the *SelectionStatus* attribute in the form of a Boolean in the Variable referenced by the *SelectionStatusVar* parameter.

Provisions of use:

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

Syntax description:

```
GetSelectionStatus    -->  Target,
                           SelectionStatusVar
Target                -->  GenericObjectReference
SelectionStatusVar    -->  ObjectReference
```

Select

If the *SelectionStatus* attribute is True, disregard this action. Otherwise, invoke the *Selection* behaviour.

Deselect

If the *SelectionStatus* attribute is False, disregard this action. Otherwise, invoke the *Deselection* behaviour.

Provision of use:

- The *Target* object shall be an available SwitchButton object.

Syntax description:

```
Deselect              -->  Target
Target                -->  GenericObjectReference
```

Toggle

If the *SelectionStatus* attribute is False, invoke the *Selection* behaviour. Otherwise, invoke the *Deselection* behaviour.

Provision of use:

- The *Target* object shall be an available SwitchButton object.

Syntax description:

```
Toggle                -->  Target
Target                -->  GenericObjectReference
```

SetLabel

Execute the following sequence of actions:

1. Set the *Label* attribute to the new value.
2. If the *Target* SwitchButton is active, redraw the SwitchButton taking into account the new value of *Label* attribute and according to its position on the DisplayStack of the active Application object.

Provision of use:

- The *Target* object shall be an available SwitchButton object.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 175 of 253

The syntax of this action is defined in the PushButton class.

48.5 Formal description

```
SwitchButton Class      --> PushButton Class,
                            ButtonStyle
ButtonStyle             --> pushbutton   |   radiobutton   |
                            checkbox
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 176 of 253

49 Action Class

Description	The Action class is a list of elementary actions that are intended to be executed synchronously. The Action class does not inherit from any MHEG-5 class; as a result, Action objects cannot be referenced individually.
Base class	None
Subclasses	None
Status	Concrete class

49.1 Attributes

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

49.1.1 Inherited attributes

This class has no inherited attributes.

49.1.2 Own exchanged attributes

This class defines the following exchanged attribute:

ElementaryActions	List of included elementary actions

49.2 Own internal attributes

This class defines no internal attribute.

49.3 Formal description

```
Action Class       -->  ElementaryAction+
ElementaryAction   -->  Activate | Add | AddItem | Append | BringToFront |
                        Call | CallActionSlot | Clear | Clone |
                        CloseConnection | Deactivate | DelItem | Deselect |
                        DeselectItem | Divide | DrawArc | DrawLine | DrawOval
                        | DrawPolygon | DrawPolyline | DrawRectangle
                        |DrawSector | Fork | GetAvailabilityStatus |
                        GetBoxSize | GetCellItem | GetCursorPosition |
                        GetEngineSupport | GetEntryPoint | GetFillColour |
                        GetFirstItem | GetHighlightStatus |
                        GetInteractionStatus | GetItemStatus | GetLabel |
                        GetLastAnchorFired | GetLineColour | GetLineStyle |
                        GetLineWidth | GetListItem | GetListSize |
                        GetOverwriteMode | GetPortion | GetPosition |
                        GetRunningStatus | GetSelectionStatus | GetSliderValue
                        | GetTextContent | GetTextData | GetTokenPosition |
                        GetVolume | Launch | LockScreen | Modulo | Move |
                        MoveTo | Multiply | OpenConnection | Preload |
                        PutBefore | PutBehind | Quit | ReadPersistent | Run |
                        ScaleBitmap | ScaleVideo | ScrollItems | Select |
                        SelectItem | SendEvent | SendToBack | SetBoxSize |
                        SetCachePriority | SetCounterEndPosition |
                        SetCounterPosition | SetCounterTrigger |
                        SetCursorPosition | SetCursorShape | SetData |
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 177 of 253

```
SetEntryPoint | SetFillColour | SetFirstItem |
SetFontRef | SetHighlightStatus | SetInteractionStatus
| SetLabel | SetLineColour | SetLineStyle |
SetLineWidth | SetOverwriteMode | SetPaletteRef |
SetPortion | SetPosition | SetSliderValue | SetSpeed |
SetTimer | SetTransparency | SetVariable | SetVolume |
Spawn | Step | Stop | StorePersistent | Subtract |
TestVariable | Toggle | ToggleItem | TransitionTo |
Unload | UnlockScreen
```

NOTE - The semantics and syntax of the elementary actions are provided earlier in the document.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 178 of 253

50 Referencing Objects, Content, Values, Colour and XYPosition

50.1 ObjectReference

Description This data type is used for referencing objects. The object referenced shall be visible to the object from which the reference is made. This means that the object shall be either an Application or Scene object, or shall be part of either the active Application or the active Scene.

The reference consists of an optional GroupIdentifier and an ObjectNumber. The default value for the GroupIdentifier is the GroupIdentifier of the Scene or Application object from which the reference was made.

```
ObjectReference          --> GroupIdentifier?,
                             ObjectNumber
GroupIdentifier          --> OctetString
ObjectNumber             --> INTEGER
```

50.2 ContentReference

Description This data type is used for referencing external sources of data. The ContentReference consists of an OctetString.

```
ContentReference         --> OctetString
```

50.3 GenericObjectReference

Description Data type that allows either direct reference to an object or indirect reference via a Variable object.

In the case of direct reference, this reference resolves to an ObjectReference directly to the target object.

In the case of indirect reference, this reference resolves to an ObjectReference to an ObjectRefVariable object. That ObjectRefVariable object shall then contain either an ObjectReference to the target object or NULL.

```
GenericObjectReference   --> DirectReference |
                             IndirectReference
DirectReference          --> ObjectReference
IndirectReference        --> ObjectReference
```

50.4 GenericContentReference

Description Data type that allows either direct reference to an external source of data or indirect reference via a Variable object.

In the case of direct reference, this reference resolves to a ContentReference directly.

In the case of indirect reference, this reference resolves to an ObjectReference to a ContentRefVariable object. That ContentRefVariable object shall then contain either a ContentReference to the content or NULL.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 179 of 253

```
GenericContentReference     --> DirectContentReference |
                                IndirectReference
DirectContentReference      --> ContentReference
IndirectReference           --> ObjectReference
```

50.5 GenericInteger

Description Data type that allows either direct inclusion of an Integer or reference to an IntegerVariable object.

```
GenericInteger              --> Value | IndirectReference
Value                       --> INTEGER
IndirectReference           --> ObjectReference
```

50.6 GenericBoolean

Description Data type that allows either direct inclusion of a Boolean or reference to a BooleanVariable object.

```
GenericBoolean              --> Value | IndirectReference
Value                       --> BOOLEAN
IndirectReference           --> ObjectReference
```

50.7 GenericOctetString

Description Data type that allows either direct inclusion of an OctetString or reference to an OctetStringVariable object.

```
GenericOctetString          --> Value | IndirectReference
Value                       --> OctetString
IndirectReference           --> ObjectReference
```

50.8 Colour

Description Data type used to specify a colour by a name - an OctetString- or by an index -an Integer referencing a palette object-.

```
Colour                      --> ColourIndex | AbsoluteColour
ColourIndex                 --> INTEGER
AbsoluteColour              --> OctetString
```

50.9 XYPosition

Description Data type used to specify the (X,Y) position in a Scene coordinate system.

```
XYPosition                  --> XPosition,
                                YPosition
XPosition                   --> INTEGER
YPosition                   --> INTEGER
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 180 of 253

50.10 Resolution of generic values

Generic values (GenericContentReference, GenericObjectReference, GenericInteger, GenericBoolean, and GenericOctetString) are used only as parameters to elementary actions. The resolution takes place when the action is invoked. As an example, consider an IntegerVariable V that is initially set to 10. If the following actions are invoked:

1. set V to 15,
2. set a Slider's *Value* to V,
3. set V to 20,

the Slider's *Value* will be set to 15. In other words, the Slider's value is set to by value, not by reference.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 181 of 253

51 Referencing MHEG-5 Objects

References to MHEG-5 objects are represented by ObjectReferences. At any time, an ObjectReference shall be resolved by taking into account both the MHEG-5 Group object that contains this reference (a Scene or an Application) and the content of the GroupIdentifier and ObjectNumber attributes of the ObjectReference.

The following presents how the ObjectReference shall be encoded according to the Group of origin and the nature of the referenced MHEG-5 object:

I. Within a Scene object,
   - A. Reference to an Ingredient of the active Scene:
     1. The GroupIdentifier may or may not be encoded.
     2. The ObjectNumber shall contain the object number of the Ingredient within the Scene.
   - B. Reference to a shared Ingredient of the active Application:
     1. The GroupIdentifier shall contain the group identifier of the active Application object.
     2. The ObjectNumber shall contain the object number of the Ingredient within the Application object.
   - C. Reference to the active Scene itself:
     1. The GroupIdentifier need not be encoded.
     2. The ObjectNumber shall be set to 0.
   - D. Reference to another Scene:
     1. The GroupIdentifier shall contain the group identifier of the Referenced Scene object.
     2. The ObjectNumber shall be set to 0.
   - E. Reference to the active Application:
     1. The GroupIdentifier shall contain the group identifier of the active Application object.
     2. The ObjectNumber shall be set to 0.
   - F. Reference to another Application object:
     1. The GroupIdentifier shall contain the group identifier of the referenced Application object.
     2. The ObjectNumber shall be set to 0.

II. Within an Application object,
   - A. Reference to an Ingredient of the Application group:
     1. The GroupIdentifier may or may not be encoded.
     2. The ObjectNumber shall contain the unique object number of the Ingredient within the Application.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 182 of 253

B. Reference to a Scene:

1. The GroupIdentifier shall contain the group identifier of the referenced Scene object.
2. The ObjectNumber shall be set to 0.

C. Reference to the Application itself:

1. The GroupIdentifier need not be encoded.
2. The ObjectNumber shall be set to 0.

D. Reference to another Application object:

1. The GroupIdentifier shall contain the group identifier of the referenced Application object.
2. The ObjectNumber shall be set to 0.

This part of ISO/IEC 13522 does not define the actual encoding of the GroupIdentifier OctetString. Every MHEG-5 application domain shall define specific forms of the GroupIdentifier.

52 Name Spaces, RemoteProgram Calls and Connections

The MHEG-5 engine has a default name space, which is the name space of the active Application object. All other Application objects that can be reached from it (by Launch or Spawn) shall also be in that name space. This name space, in a wide sense, also should include enough information to perform remote calls to named methods through the MHEG-5 Program objects.

NOTE - In the Annex C (informative), an explanation may be found on how the MHEG-5 engine attaches to this name space.

However, it is also possible for an MHEG-5 application to connect to another name space temporarily, using the OpenConnection action. That new name space may be used for named method calls (through the MHEG-5 Program objects) and in order to access an MHEG-5 Scene object that is located in another name space than its Application object. The following rule applies to the references to and from such a Scene object:

1. The reference to a Scene object in another name space than the Application object shall be encoded within a TransitionTo action of which the ConnectionTag parameter indicates the connection with the entity that administers that other name space.
2. All ContentReferences from that Scene object shall be interpreted in the «other» name space.
3. All GroupIdentifier references from that Scene object shall be interpreted in the default name space, except as indicated by rule 1. above.

Reminder: whether or not the MHEG-5 engine shall implement the OpenConnection and CloseConnection actions is a decision of the application domain.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 183 of 253

53 Event Handling

53.1 Types of events

This part of ISO/IEC 13522 defines the following events: *IsAvailable, ContentAvailable, IsDeleted, IsRunning, IsStopped, TestEvent, UserInput, TimerFired, AsynchStopped, InteractionCompleted, TokenMovedFrom, TokenMovedTo, FirstItemPresented, LastItemPresented, HeadItems, TailItems, ItemSelected, ItemDeselected, StreamEvent, StreamPlaying, StreamStopped, CounterTrigger, HighlightOn, HighlightOff, CursorEnter, CursorLeave, AnchorFired, IsSelected, IsDeselected, EntryFieldFull, EngineEvent.*

An event always emanates from one specific object, called the event source. The semantics of each MHEG-5 class define the circumstances under which an object of that class generates a specific event.

Some of the event types above have associated with them a data value. That value is used in order to determine whether the associated Link should fire, as described below. The following table lists the data value associated with each event type:

Event Type	Associated data	Associated value type
AnchorFired	*AnchorTag*	*OctetString*
CounterTrigger	*Identifier*	*INTEGER*
EngineEvent	*EventTag*	*INTEGER*
FirstItemPresented	*Index*	*BOOLEAN*
HeadItems	*Number of items*	*INTEGER*
ItemDeselected	*Index*	*INTEGER*
ItemSelected	*Index*	*INTEGER*
LastItemPresented	*Index*	*BOOLEAN*
StreamEvent	*StreamEventTag*	*OctetString*
TailItems	*Number of items*	*INTEGER*
TestEvent	*TestResult*	*BOOLEAN*
TimerFired	*TimerIdentifier*	*INTEGER*
TokenMovedFrom	*Index*	*INTEGER*
TokenMovedTo	*Index*	*INTEGER*
UserInput	*UserInputEventTag*	*INTEGER*
All others	None.	N/A

Events which types have associated data must always be generated with that data. E.g., when a *TimerFired* event is generated, it must be accompanied by the *TimerIdentifier*.

53.2 Synchronous events and asynchronous events

Events may occur for two reasons.

1. A process that is asynchronous to the MHEG-5 engine produces an event.

   The resulting event is then called asynchronous. Asynchronous events are all events of the types *AnchorFired, AsynchStopped, ContentAvailable, CounterTrigger, CursorEnter, CursorLeave., EngineEvent, EntryFieldFull, InteractionCompleted, StreamEvent, StreamPlaying, StreamStopped, TimerFired and UserInput.*

2. The event is the direct result of the execution of an elementary action.

   The resulting event is then called synchronous. Synchronous events are all events of the types *FirstItemPresented, HeadItems, HighlightOff, HighlightOn, IsAvailable, IsDeleted, IsDeselected, IsRunning,*

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 184 of 253

IsSelected, IsStopped, ItemDeselected, ItemSelected, LastItemPresented, TailItems, TestEvent, TokenMovedFrom and TokenMovedTo. The semantics of the classes of this part of ISO/IEC 13522 explicitly state when such an event is generated.

53.3 Event handling and Links

Every MHEG-5 Link has a LinkCondition and a LinkEffect. When the MHEG-5 engine examines an event that has occurred, it shall check all active Links (of the active Application and Scene objects) to see if their EventType and EventSource attributes match the type and source of the event in question. For each of the Links that fulfil this condition, the associated data of the event is checked against the optional EventData attribute of the Link. Links that fulfil this condition as well (or that fulfil it by default since they have no EventData) are said to be *fired.*

The MHEG-5 engine is driven by the occurrence of asynchronous events. At the occurrence of an asynchronous event, the MHEG-5 engine shall examine all active Links of the active Application and Scene objects to determine if they have fired. For each of the fired Links, the elementary actions of its LinkEffect shall be stored in a queue for sequential execution. This part of ISO/IEC 13522 does *not* specify the order in which two Links that fire on the same event are to be handled.

Default error handling is the following: if one of the aforementioned elementary action produces an error, that elementary action shall be ignored. It is however allowed for an application domain to use EngineEvent events to indicate the error situation to the MHEG-5 application (e.g., to send to an error message).

As a direct result of a LinkEffect being executed, synchronous events may occur. These events shall be dealt with directly by the MHEG-5 engine. In other words, after the execution of each elementary action, the MHEG-5 engine shall check if any additional Links have fired as the result of a synchronous events occurring. If this is the case, that Link and all its effects shall be completely processed before the MHEG-5 engine continues to process the next elementary action of the original Link.

Any asynchronous events (such as a UserInput event) that occur while the original asynchronous events are being processed are not dealt with until after the completion of the entire process above. The asynchronous events which are not handled, are queued.

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" actions. If such an action occurs, the pending asynchronous events whose origin is after the context switch shall be removed from the asynchronous event queue. Elementary actions waiting for execution shall be removed from the action queue as well.

In this context, it should be noted that some MHEG-5 actions, such as Run, have an effect that continues after the completion of the action itself. For example, when a Bitmap object is run, the Run action returns as soon as the Bitmap has been displayed, thus allowing the Link processor to continue its work. The effect of the action (i.e., that the bitmap is on the screen) still continues after the completion of the action.

Another important aspect is that it is possible for a Link to deactivate itself in its LinkEffect. Such an action shall be postponed until the LinkEffect has been completely executed.

53.4 User input

This part of ISO/IEC 13522 specifies neither the user input devices nor the types of events that are generated. The «raw» user input received by the MHEG-5 engine (for instance in the form of remote control commands) shall be translated by the engine, where appropriate, into occurrences of *UserInput* events. These events all have a tag, which specifies which *UserInput* event has occurred. The tag is an Integer; the semantics of the tag are defined by a InputEventRegister attached to the Scene object.

Once the MHEG-5 engine has translated the raw user input into one or more *UserInput* events, that or those events are treated as asynchronous events by the engine as described above.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 185 of 253

53.5 User interaction

MHEG-5 objects that belong to the class Interactible may be in a certain state, called «interacting», which is signalled by the *InteractionStatus* attribute of the object being True. When an object is in this state, no *UserInput* events shall be generated by the MHEG-5 engine. The reason for this rule is that the MHEG-5 engine might need to use the «raw» user input in order to interact with the Interactible, and therefore may not be able to generate the normal UserInput events.

At all times, at most one object shall be interacting.

However, all other events are still generated. This makes it possible, for instance, to implement time-outs.

53.6 Cursor Events

CursorLeave and *CursorEnter* events defined earlier need the following precision. It may occur that a cursor moves from one area defined by an active Interactible immediately to another area defined by another active Interactible; that may be the case, for instance, when one area is on top of the other, that is when one area is above the other in the DisplayStack and they overlap. In such case, a *CursorLeave* event shall be generated by the former Interactible (the topmost of the two in the DisplayStack), followed by a *CursorEnter* event generated by the latter Interactible (the lower one in the DisplayStack).

Cursor Events have also to be generated if an active Interactible is moved beneath the cursor position, or if an Interactible beneath the cursor position becomes active, or if the stacking order of Interactibles beneath the cursor position changes in such a way, that the topmost Interactible beneath the cursor position is changed.

In general, only one active Interactible per time can have the cursor in its bounding box. This has to be reflected by appropriate sequences of CursorLeave and CursorEnter events.

Finally, it should also be recalled here that an inactive Interactible shall not generate any cursor events.

53.7 Error Handling

In general, default error mechanism of the MHEG-5 engine consists in ignoring the cause of the error and continuing to the next step. For instance if an interchanged object is an instance of a class that the MHEG-5 engine does not recognise, the object shall be ignored. Any subsequent action that may occur in relation with this object and that produces an error shall be ignored. More generally, any cause of error (e.g. in an elementary action) shall result in ignoring that cause (e.g. the elementary action).

However, as mentioned earlier, it is allowed for an application domain to use EngineEvent events to indicate the error situation to the MHEG-5 application (e.g., to send to an error message).

54 Rendering Visibles

54.1 Coordinate system

This subclause describes the semantics associated with displaying objects of classes that inherit from the Visible class. The MHEG-5 engine shall have access to exactly one display, which is called the «screen» in the rest of this subclause. The screen is an orthogonal coordinate system, with an x axis from left to right and a y axis from top to bottom. The origin is in the top left corner. Depending on the SceneCoordinateSystem and AspectRatio attributes of the currently active Scene object, the coordinate system may have different sizes and aspect ratios.

54.2 Bounding box

Each Visible has a bounding box, specified by the internal attributes Position and BoxSize. These are both expressed in the Scene coordinate space. The contents of the Visible shall be anchored to the top left corner

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 186 of 253

of the bounding box. Furthermore, the contents of the Visible shall be clipped so that no parts that fall outside the bounding box are rendered. The clipping shall be performed in the following fashion:

- For Text and its subclasses, the clipping is performed in such a way that the positioning of the text within the bounding box is respected. Here, only the case where LineOrientation is *horizontal* is explained; the *vertical* case is obtained analogously.
  - Horizontal justification:

  If the Text object has a TextWrapping attribute set to True, its content is split into as many lines as needed in order to fit into the bounding box. In addition, if the HorizontalJustification attribute of the Text object is set to *justified*, the lines are displayed so that their beginning is aligned with the left edge of the bounding box and that their end is aligned with the right edge of the bounding box.

  If the TextWrapping attribute is set to False, the following rules apply for horizontal justification:

  If HorizontalJustification is set to *start*, the lines are clipped so that the beginning of each line is aligned with the left edge of the bounding box.

  If HorizontalJustification is set to *end*, the lines are clipped so that the end of each line is aligned with the right edge of the bounding box.

  If HorizontalJustification is set to *centre*, the lines are clipped on both sides, so that the centre of each line is aligned with a virtual vertical line at the centre of the bounding box.

  - Vertical Justification:

  For a Text that has the VerticalJustification attribute set to *end*, its content is clipped so that the end of the text is aligned with the bottom of the bounding box; thus, the beginning of the text is obscured. Conversely, if VerticalJustification is set to *start*, the text is clipped so that its end is obscured. Finally, if VerticalJustification is set to *centre*, is clipped so that the centre of the text is aligned with the centre of the bounding box, meaning that both beginning and end are obscured.

  - Background Colour:

  If the text object has a background colour other than transparent, its entire bounding box shall be filled with that colour, whether or not the text fills the bounding box.
- For Slider, Rectangle, and subclasses of Button, the object is not clipped but rather scaled to fit exactly within its bounding box.
- Other Visibles are clipped so that their top left corner is positioned at the top left corner of the bounding box. For these Visibles, some part of the bounding box may be empty; that part shall then be transparent.

54.3 Display stack

The MHEG-5 engine shall implement a display stack. Each Visible is initially rendered according to its position in the *DisplayStack* attribute of the current Application object. Visibles may be moved up and down in the display stack using the actions BringToFront, SendToBack, PutBefore and PutBehind. When a Visible is moved in this way, (parts of) other Visibles may be obscured or uncovered. The display shall then be updated accordingly.

54.4 Transparent objects

Transparency of objects are specified with an Integer in the range [0, 100] (percent). Interpreting the integer between 1 and 99 depends on the implementation and this part of ISO/IEC 13522 does not define it. Neither the actual rendering and display accuracy, nor the algorithms used to perform these tasks are specified by this part of ISO/IEC 13522. However, transparency values of 0% and 100% are specified; they are defined as follows.

Objects of 0% transparency shall be rendered as non transparent objects, i.e., the original colours shall be rendered without any modification. Objects of 100% transparency shall be rendered as completely transparent objects, i.e., the original colours shall not be rendered, and the background shall be perceived.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 187 of 253

I. For each location in the Scene coordinate space, at most the *N* objects whose bounding boxes contain that location are considered. These objects are ordered in the so-called display stack. Let's call the objects O_1, O_2, [...], O_N, where O_N denotes the topmost object.

II. The computation of the final pixel value is done from the bottom of the stack. The pixel value after taking into account *j* objects is called V_j. The final pixel value, therefore, is called V_N.

III. The computation of the pixel value starts from a completely black background. Therefore, V_0 = <black>.

IV. For a *j* taking successively the values 1 through *N*, the following is performed:

A. If the pixel value of O_j to be considered is called P_j and is *p* percent translucent:

$V_j = V_{j-1} (p/100) + P_j (1 - p/100)$;

This formula may be applied to all colour components of a pixel if the colour space is not based on the luminance and colour difference signals (RGB is such a case). However, this formula must be applied only to the luminance of a pixel if the colour space is based on the luminance and colour difference signals such as YUV. In addition, the values in this formula are all assumed to be expressed in one and the same linear colour space. If a non linear colour space is used, this formula may be taken into account for the precise rendering.

B. specifically, if the pixel is completely transparent (invisible):

$V_j = V_{j-1}$;

C. specifically, if the pixel is completely opaque:

$V_j = P_j$.

This model shall not be understood to mandate any specific rendering and/or display process in a specific implementation of an MHEG-5 engine. Its sole purpose is to define unambiguously what the semantics are of combining different transparent graphics objects. Neither the actual rendering and display accuracy, nor the algorithms used to perform these tasks are defined by this part of ISO/IEC 13522.

54.5 Pixel aspect ratio

Generally the pixel aspect ration of a Scene and its content are assumed to be the same, a pixel of the Scene's content is then exactly mapped to a pixel of the Scene, no scaling is needed. If however the two aspect ratios are different, the MHEG-5 engine may scale the content - taking into account the two aspect ratios - in order to compensate for the presentation of the content. Note that this scaling is up to the implementation of the MHEG-5 engine and is an optional feature; this part of ISO/IEC 13522 does not define any specific way to do it.

Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 188 of 253

Annex A

(normative)

ASN.1 notation

This annex describes the ASN.1 notation for the syntax of MHEG-5 objects conforming to ISO/IEC 13522-5.

The encoding of the MHEG-5 objects from this ASN.1 syntax shall make use of the *Distinguished Encoding Rules (DER)* defined in ISO/IEC 8825-1:1995 | ITU-T Recommendation X.690. The alternative encoding is the textual notation defined in Annex B.

The syntax that shall be used in the DER is detailed below.

```
--$PREFIX=ISOMHEG-mheg-5:mheg-5
-- Module: mheg-5

ISO13522-MHEG-5
{joint-iso-itu-t(2) mheg(19) version(1) mheg-5(17)} DEFINITIONS IMPLICIT
TAGS ::=
BEGIN

-- This module defines the MHEG-5 abstract syntax which consists of data values of type:
--  ISO13522-MHEG-5.InterchangedObject.
-- This abstract syntax is identified by the name: {joint-iso-itu-t(2) mheg(19) version(1) mheg-5(17)}.

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

-- A.1 Root Class ____________________________________________________

RootClass ::= ObjectReference

-- A.2 Group Class ___________________________________________________

GroupClass ::= SET
{
 RootClass (WITH COMPONENTS
  {external-reference (WITH COMPONENTS {..., object-number (0)}) PRESENT,
   internal-reference ABSENT}),
 standard-identifier [2] StandardIdentifier OPTIONAL,
 standard-version [3] INTEGER (1) OPTIONAL,
 object-information [4] OCTET STRING OPTIONAL,
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 189 of 253

```
  on-start-up [5] ActionClass OPTIONAL,
  on-close-down [6] ActionClass OPTIONAL,
  original-group-cache-priority [7] INTEGER (0..255) DEFAULT 127,
  items [8] SEQUENCE SIZE (1..MAX) OF GroupItem OPTIONAL
}

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

GroupItem ::= CHOICE
{
  resident-program [9] ResidentProgramClass,
  remote-program [10] RemoteProgramClass,
  interchanged-program [11] InterchangedProgramClass,
  palette [12] PaletteClass,
  font [13] FontClass,
  cursor-shape [14] CursorShapeClass,
  boolean-variable [15] BooleanVariableClass,
  integer-variable [16] IntegerVariableClass,
  octet-string-variable [17] OctetStringVariableClass,
  object-ref-variable [18] ObjectRefVariableClass,
  content-ref-variable [19] ContentRefVariableClass,
  link [20] LinkClass,
  stream [21] StreamClass,
  bitmap [22] BitmapClass,
  line-art [23] LineArtClass,
  dynamic-line-art [24] DynamicLineArtClass,
  rectangle [25] RectangleClass,
  hotspot [26] HotspotClass,
  switch-button [27] SwitchButtonClass,
  push-button [28] PushButtonClass,
  text [29] TextClass,
  entry-field [30] EntryFieldClass,
  hyper-text [31] HyperTextClass,
  slider [32] SliderClass,
  token-group [33] TokenGroupClass,
  list-group [34] ListGroupClass
}

--A.3. Application Class ____________________________________________
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 190 of 253

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

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

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

-- A.4 Scene Class ---------------------------------------------------

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

SceneCoordinateSystem ::= SEQUENCE
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 191 of 253

```
{
 x-scene INTEGER,
 y-scene INTEGER
}

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

NextScene ::= SEQUENCE
{
 scene-ref OCTET STRING,
 scene-weight INTEGER (0..255)
}
-- A.5. Ingredient Class ____________________________________________

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

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

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

-- A.6. Link Class _____________________________________________

LinkClass ::= SET
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 192 of 253

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

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

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


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 193 of 253

```
  item-deselected(29),
  entry-field-full(30),
  engine-event(31)
}

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

-- A.7 Program Class ____________________________________________________

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

-- A.8 Resident Program Class ___________________________________________

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

-- A.9 Remote Program Class _____________________________________________

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

-- A.10 Interchanged Program Class ______________________________________

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

-- A.11 Palette Class ___________________________________________________

PaletteClass ::= IngredientClass
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 194 of 253

```
 (WITH COMPONENTS
  {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- A.12 Font Class ---------------------------------------------------

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

-- A.13 Cursor Shape -------------------------------------------------

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

-- A.14 Variable Class -----------------------------------------------

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

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

-- A.15 Boolean Variable Class ---------------------------------------

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

-- A.16 Integer Variable Class ---------------------------------------

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


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 195 of 253

```
-- A.17 Octet String Variable Class _____

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

-- A.18 Object Reference Variable Class _____

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

-- A.19 Content Reference Variable Class _____

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

-- A.20 Presentable Class _____

PresentableClass ::= IngredientClass

-- A.21 Token Manager Class _____

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

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

-- A.22 Token Group Class _____

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

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


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 196 of 253

```
}

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

-- A.23 List Group Class ---------------------------------------------

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

-- A.24 Visible Class ------------------------------------------------

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

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

-- A.25 Bitmap Class -------------------------------------------------

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


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 197 of 253

```
-- A.26 Line Art Class ____________________________________________

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

-- A.27 Rectangle Class ___________________________________________

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

-- A.28 Dynamic Line Art Class ____________________________________

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

-- A.29 Text Class ________________________________________________

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

Justification ::= ENUMERATED
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 198 of 253

```
{
 start(1),
 end(2),
 centre(3),
 justified(4)
}

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

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

-- A.30 Stream Class ---------------------------------------------------

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

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

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

-- A.31 Audio Class ----------------------------------------------------

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


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 199 of 253

```
 original-volume [99] INTEGER DEFAULT 0
}

-- A.32 Video Class ____________________________________________

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

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

-- A.33 RTGraphics Class _______________________________________

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

-- A.34 Interactible Class _____________________________________

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

-- A.35 Slider Class ___________________________________________

SliderClass ::= SET
{
 COMPONENTS OF VisibleClass
  (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
 COMPONENTS OF InteractibleClass,
 orientation [102] Orientation,
 max-value [103] INTEGER,
 min-value [104] INTEGER DEFAULT 1,
```


Licensed to Immanuel Freedman. ANSI store order # X_671658. Downloaded 08/25/2020. Single user license only. Copying and networking prohibited.

Ex. 1026 – TCL Indus Holdings Co., Ltd.
TCL Indus Holdings Co., Ltd. v. LGE – IPR2023-00238
Page 200 of 253