



INTERNATIONAL STANDARD ISO/IEC 14496-20:2006
TECHNICAL CORRIGENDUM 1

Published 2007-02-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Coding of audio-visual objects —
Part 20:
Lightweight Application Scene Representation (LAsE_R) and
Simple Aggregation Format (SAF)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 20: Représentation de scène d'application allégée (LAsE_R) et format d'agrégation simple (SAF)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 14496-20:2006 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

In subclause 6.2.2 “Fonts”, replace the first paragraph with:

LASeR supports the encoding of fonts. Fonts may be encoded separately from the scene using ISO/IEC 14496-18, and sent as a media stream together with the scene stream. Fonts may also be transmitted using SVG elements related to font encoded using the `privateAnyXMLElement` SDL construct, as specified in 12.2.1 and 12.2.3.

In subclause 6.3 “Timing Model”, replace the first paragraph with the following:

This subclause describes the notion of Scene Times, Media Times, and Encoded Scene Times.

In subclause 6.3, replace the following paragraph:

Scene times within that access unit (“begin” in this example) are encoded for transmission relative to the scenetime of the access unit. In this example, the “begin” time X is transmitted as the Encoded Scene Time X-ST(r). For LASeR commands inside a script element, ST(r) is the scene time when the script is activated.

with:

For LASeR commands inside a conditional element, ST(r) is the scene time when the conditional is activated.

In the last paragraph of subclause 6.3, replace sentence:

The number of ticks per seconds for time values relating to the scene time line is defined by the timeResolution attribute of the LASeRHeader.

with:

The number of ticks per seconds and origin of time values relating to the scene time line are defined by the timeResolution and timeEncoding attributes of the LASeRHeader.

In the last paragraph of subclause 6.3, remove sentence:

Attributes “begin” and “end” are encoded as offset from the scene time of the current access unit.

In subclause 6.5 “Supported Events”, replace in Table 1:

| | | | | |
|----------|---------------------------|---|----|----|
| “pause” | urn:mpeg:mpeg4:laser:2005 | Freezes the clock of the timed object they are sent to, and have no effect on non timed objects. | No | No |
| “resume” | urn:mpeg:mpeg4:laser:2005 | Restarts the clock of the timed object they are sent to, and have no effect on non timed objects. | No | No |

with:

| | | | | |
|---------|---------------------------|---|----|----|
| “pause” | urn:mpeg:mpeg4:laser:2005 | Freezes the clock of the target timed element, and have no effect on non timed elements. | No | No |
| “play” | urn:mpeg:mpeg4:laser:2005 | If the target timed element is not active, receiving this event starts the timed element as if it received an activate event. If the timed element has been paused, receiving this event resumes the clock of the timed element. It have no effect on non timed elements. | No | No |

In subclause 6.5, add to Table 1:

| | | | | |
|-----------------|---------------------------|--|----|----|
| “executionTime” | urn:mpeg:mpeg4:laser:2005 | This pseudo event can only appear in time expressions in Replace Attribute and Insert Indexed Attribute commands; upon the execution of the command, it is replaced by the current time. | No | No |
|-----------------|---------------------------|--|----|----|

In subclause 6.5, remove Table 2 and the associated introductory sentence.

In subclause 6.6.2.1 “LASeR headers semantics”, replace the following:

LASeR defines a way to partition scenes into incremental scene segments, allowing services to be built of different scene segments, the first scene segment containing a NewScene update, the other scene segments having the **append** bit set and not starting with a **NewScene** update, and being designed as addition to the first scene segment.

LASeR defines an interface to persistent storage. The LASeR engine shall cache permanent streams and selected scene information on a best effort basis. The principles behind this caching closely follow the state caching mechanism in HTTP, commonly called cookies [RFC 2965]. Within the LASeR streams, there are two commands that may be used. One command saves, associated with a string name called **groupID**, the values of some attributes of some nodes. This storage is scoped by the domain-name and path computed from the source using the fields in the LASeR header. The other command restores the attributes (if any) previously saved under the given **groupID**, as scoped by the domain-name and path.

The stored values and permanent streams are scoped by the domain-name and path. That is, it is possible for both ".acme.com" and ".widget.com" to store data under the same **groupID** and for that storage to be distinct; similarly for "/user/laser-expert/" and "/user/laser-novice/" at ".acme.com" to save state under the same **groupID**, and for those saved states to be distinct.

with:

LASeR defines a way to partition scenes into incremental scene segments, allowing services to be built of different scene segments, the first scene segment containing a NewScene update, having the **newSceneIndicator** bit set, and the other scene segments not starting with a **NewScene** update, and being designed as addition to the first scene segment.

LASeR defines an interface to persistent storage. The LASeR engine has the ability to cache selected scene information, using specific LASeR commands, on a best effort basis. The principles behind this caching closely follow the state caching mechanism in HTTP, commonly called cookies [RFC 2695]. Within the LASeR streams, there are two commands that may be used. One command saves, associated with a string name called **groupID**, the values of some attributes of some nodes. This storage is scoped by the domain-name and path computed from the source using the fields in the LASeR header. The other command restores the attributes (if any) previously saved under the given **groupID**, as scoped by the domain-name and path.

The stored values are scoped by the domain-name and path. That is, it is possible for both ".acme.com" and ".widget.com" to store data under the same **groupID** and for that storage to be distinct; similarly for "/user/laser-expert/" and "/user/laser-novice/" at ".acme.com" to save state under the same **groupID**, and for those saved states to be distinct.

In subclause 6.6.2.2 "Attributes of LASeR Header", do the following edits:

- Remove Table 3

- Remove the bullet defining "append"

- Remove the bullet defining "hasStringIDs"

- Add the following 2 bullets:

- **newSceneIndicator**: this Boolean indicates if this scene segment starts with a NewScene command or a RefreshScene command. When it is set, the first LASeR command in this scene segment shall be a NewScene or a RefreshScene; otherwise, the first LASeR command in this scene segment shall be neither a NewScene nor a RefreshScene.

NOTE : the terminal may remove (parts of) the existing scene tree, for resource optimization, when it receives a LASeRHeader with this Boolean set to true.

- **timeEncoding**: this attribute defines the reference for encoded times: if the value is 0, all times in this segment are relative to the scene time line; if the value is 1, all times in this segment are relative to the beginning of this segment; if the value is 2, all times are encoded in the same way as in the previous segment.

In subclause 6.6.2.4 "Encoding Context", add the following sentence:

Binary IDs shall not be referred to across scene segments.

Create subclause 6.6.3 "String IDs Encoding":

6.6.3 String IDs Encoding

The element StringIDTable is an optional first child of the LAsERUnit in LAsERML. This StringID table informs the encoder to encode the given IDs as a string ID. When hasStringIDs is set, this table can be omitted as all IDs shall be encoded as strings.

```
<StringIDTable strings="<oneString> <oneString> ..."/>
```

An ID that must be encoded as a string ID must be declared at the latest in the StringIDTable of the LAsERUnit where it is first used (by definition or by reference). When a LAsER AccessUnit is a random access point (RAP), it shall contain a table containing all already known string IDs.

In subclause 6.7.1 "LAsER Scene Commands Overview", replace the second bullet of the second paragraph with the following:

The following attributes cannot be updated: attributeName, id, type, xml:space, preserveAspectRatio, the x and y attributes of the text element and the following attributes of the animation elements: by, from, to, values and fill. This constraint can be worked around by updating the whole element.

Append the following at the end of subclause 6.7.1:

LAsER Commands modify a scene tree regardless of their representation, transport or encoding and regardless of the origin of the scene tree on which they apply. Commands and initial scene may be represented separately in different documents, encoded and delivered in different streams.

In subclause 6.7.2.1 "Add attributes semantics", add the following paragraph at the end of the subclause:

The value designated by **operandElementId** and **operandAttributeName** is the presentation value. The following attributes cannot be added to: **children, d, mpath, begin, end**.

In subclause 6.7.4.2 "Delete Attributes", replace:

- **attributeName**: this attribute defines the name of the attribute in which the deletion happens, by default "children". This attribute is only allowed in conjunction with index.

with:

- **attributeName**: when index is specified, this attribute defines the name of the attribute in which the deletion happens, by default "children". When index is not specified, the whole attribute is removed.

In subclause 6.7.5.1 "Insert Semantics", replace:

The **Insert** command inserts an element in a parent list.

with:

The **Insert** command inserts an element or value in a list.

In the examples, replace "Insert href" with "Insert ref" three times.

In subclause 6.7.5.2 "Insert attributes", add the following sentence at the end of the second bullet:

In the case of indexed inserts to the begin/end attributes of all timed elements, the index is ignored, which means the insertion will be done at the end of the list. Indexed inserts to the attributes of type path, namely d and mpath, are not allowed.

In subclause 6.7.8.1 "Replace Semantics", replace the third paragraph with the following:

The **Attribute Replacement** command replaces a specific attribute of an element with a new value. The **ref** attribute specifies the parent element, the **attributeName** attribute specifies which attribute of the parent

element is replaced, and the **value** attribute or the content of the Replace element specifies the new value. If **index** is specified, then the replacement is done on the value with rank **index** in the list.

When replacing begin and end attributes, the pseudo-event executionTime can be used to signify the time at which the command is executed. After execution, this pseudo-event is replaced by its time value.

Insert the following two paragraphs at the end of subclause 6.7.8.1:

The following restrictions apply:

- attributeName may point at a private XML attribute, in which case index is not allowed.
- Indexed replacements to the begin/end attributes of all timed elements are not allowed.
- Indexed replacements to the attributes of type path, namely d and mpath, are not allowed.
- The values **id**, **begin**, **end**, **d** and **mpath** are not allowed for operandAttributeName.

The value designated by **operandElementId** and **operandAttributeName** is the presentation value as defined in the SVG specification.

Attribute replacement with attributeName="textContent" and a value of index replaces the index-th component of the text content of the target element, if it exists. Without an index value, the text content is flattened and replaced as a whole, as in text editing. In the absence of text content, this command is ignored.

In subclause 6.7.8.2 "Replace Attributes", replace the semantics of the attributeName attribute with the following:

- **attributeName**: this attribute defines the name of the replaced attribute. Acceptable values are the name of any attribute of a LAsER element or the name of any private XML attribute used in the scene.

In subclause 6.7.10.2 "Save Command Attributes", in the semantics of the attribute field, add the following sentence.

The following attributes are not allowed in the **attributes** list: **children**, **begin**, **end**.

In subclause 6.7.11.2 "SendEvent Command Attributes", add at the end:

The above attributes carry information from the interface of the events, as specified in the next table:

| Name of event | Attribute of event | Attribute of SendEvent |
|---------------|--------------------|---|
| TimeEvent | detail | intvalue |
| UIEvent | detail | intvalue |
| WheelEvent | wheelDelta | intvalue |
| MouseEvent | screenX | <i>not carried</i> |
| | screenY | <i>not carried</i> |
| | clientX | pointvalue |
| | clientY | pointvalue |
| | button | intvalue |
| TextEvent | data | stringvalue |
| KeyEvent | keyIdentifier | stringvalue when unknown keyIdentifier, intvalue when a LAsER key code is defined |
| ProgressEvent | lengthComputable | <i>not carried</i> |
| | loaded | intvalue |
| | total | <i>not carried</i> |

In subclause 6.8.1 "Conventions", remove duplicated sentence:

The list of possible attributes for an element is given in the summary table in subclause 6.8.41 and fix reference.

In subclause 6.8.3 SVG a, add paragraphs at the end:

The expression "linked content" is used in the definition of the target attribute in [W3C SVG11]. For LAsER content, if the incoming data starts with a NewScene, "linked content" is the scene described by the incoming data; if the incoming data starts with any other command than a NewScene or RefreshScene, "linked content" is the current scene modified by the commands contained in the incoming data.

When an a element is activated, the incoming data replaces the previously incoming LAsER Commands and media.

In subclause 6.8.4.2 "SVG animate Attributes", in the semantics of the enabled attribute, add the following sentence:

The default value for this attribute is 'true'.

In subclause 6.8.5.2 "SVG animateColor Attributes", in the semantics of the enabled attribute, add the following sentence:

The default value for this attribute is 'true'.

In subclause 6.8.6.2 "SVG animateMotion Attributes", in the semantics of the enabled attribute, add the following sentence:

The default value for this attribute is 'true'.

In subclause 6.8.7.2 "SVG animateTransform Attributes", in the semantics of the enabled attribute, add the following sentence:

The default value for this attribute is 'true'.

In subclause 6.8.8.2 "SVG audio Attributes", add this sentence to bullet about clipBegin:

The value of clipBegin is taken into account when the media element is activated.

Add this sentence to bullet about clipEnd:

The value of clipEnd is taken into account when the media element is activated.

Insert a new subclause between 6.8.9 and 6.8.10:

6.8.10 LAsER conditional

6.8.10.1 Semantics

The **LAsER conditional** element allows sets of scene commands to be inserted in the scene, for later execution upon activation by time or through the **XML Events listener** element.

6.8.10.2 Attributes

- **begin**: this attribute specifies the time at which the conditional element is triggered. This attribute is not animatable and not inheritable. The default value is indefinite.
- **enabled**: this Boolean attribute specifies whether the element is activatable or not. This attribute is not animatable and not inheritable. The default value is true.

Replace subclause 6.8.10 "SVG Cursor" with the following:

6.8.11 LAsER cursorManager

6.8.11.1 Semantics

The LAsER cursorManager element may point to any LAsER element to confer it the semantics of virtual pointer. The virtual pointer may be moved through LAsER Commands or other interaction by moving the object associated with the virtual pointer. When moved in and out of mouse-sensitive elements, the same events shall be generated as if a pointing device such as a mouse had been moved in its stead. For example:

- when the virtual pointer is moved in a mouse-sensitive region, a mouseover is generated,
- when the virtual pointer is moved out of a mouse-sensitive region, a mouseout is generated,
- when the virtual pointer is moved, a mousemove is generated,
- when the virtual pointer is located on a mouse-sensitive region and a activate event is received by the virtual pointer, the activate event is translated to a click event sent to the region.

LAsER elements implementing the interaction moving the object associated with the virtual pointer shall be placed as children of the cursor element.

Example:

```
<lsr:cursorManager id="cursorManager1" xlink:href="#vp">
  <!-- pointer movement -->
  <ev:listener event="accessKey(UP)" handler="#s1"/>
  <lsr:conditional id="s1">
    <lsr:Add ref="g" attribute="translation" pointvalue="0 -5"/>
  </lsr:conditional>
  <ev:listener event="accessKey(DOWN)" handler="#s2"/>
  <lsr:conditional id="s2">
    <lsr:Add ref="g" attribute="translation" pointvalue="0 5"/>
  </lsr:conditional>
  <ev:listener event="accessKey(LEFT)" handler="#s3"/>
  <lsr:conditional id="s3">
    <lsr:Add ref="g" attribute="translation" pointvalue="-5 0"/>
  </lsr:conditional>
  <ev:listener event="accessKey(RIGHT)" handler="#s4"/>
  <lsr:conditional id="s4">
    <lsr:Add ref="g" attribute="translation" pointvalue="5 0"/>
  </lsr:conditional>
</lsr:cursorManager>
<!-- activate event to be translated to a click -->
<ev:listener event="accessKey(FIRE)" handler="#cursorManager1"/>
<!-- mouse sensitive shape -->
<polygon ...>
  <ev:listener event="click" handler="#someConditional"/>
</polygon>
<g id="vp" fill="black">
  <!-- crosshair -->
  <line x1="-5" y1="0" x2="5" y2="0"/>
  <line x1="0" y1="-5" x2="0" y2="5"/>
</g>
```

6.8.11.2 Attributes

Attributes of lsr:cursorManager are x, y and xlink:href as defined on the cursor element of SVG 1.1 [W3C SVG11]

Replace subclause 6.8.15 "SVG g element" with the following:

The **SVG g** element is described in subclause 5.2 of [W3C SVG11].

And create the following subclauses:

6.8.28 LAsER rectClip

6.8.28.1 Semantics

The **LAsER rectClip** element acts as a clipping element, limiting the rendering of its children to a device-pixel-aligned rectangle, in addition to the semantics of the **SVG g** element. The rectangle is not sensitive to rotation or skew of the local coordinate system, and if scaled, its size is rounded to the nearest pixel. If the **rectClip** element carries a transform attribute, the corresponding transformation is applied to the **rectClip** children but not to the clipping rectangle itself.

6.8.28.2 Attributes

- **size**: a pair of coordinates defining the width and height of the clipping rectangle. All children are clipped by this axis-aligned rectangle centered on the origin of the local coordinate system and of size (size.x, size.y). This attribute is animatable but not inheritable.

6.8.30 LAsER selector

6.8.30.1 Semantics

The **LAsER selector** element acts as a selection element, rendering zero or one of its children, in addition to the semantics of the **SVG g** element.

6.8.30.2 Attributes

- **choice**: the rendering mode selector. It is either one of the value 'none' or 'all' or an integer representing the 0-based index of the child to be displayed. This attribute is animatable but not inheritable.

6.8.32 LAsER simpleLayout

6.8.32.1 Semantics

The **LAsER simpleLayout** element acts as a simple layout tool, by spacing its children by a specified amount, thus creating rows or columns of children, in addition to the semantics of the **SVG g** element.

6.8.32.2 Attributes

- **delta**: a pair of coordinates defining the spacing between children. The first child is displayed at (0,0) of the local coordinate system, and the n-th child is displayed at ((n-1)*size.x,(n-1)*size.y) of the local coordinate system. One of the two coordinates shall be 0. This creates a row or column of objects. This attribute is animatable but not inheritable.

NOTE If a child is hidden, the effect will be that a gap is visible between the previous and following child.

In subclause 6.8.19.2 "XML Events listener Attributes", do the following edits:

- in the semantics of the **enabled** attribute, add the following sentence.

The default value for this attribute is 'true'.

- remove the semantics of **timeAttribute** and **delay**.

- replace the semantics of the **handler** attribute with the following:

- **handler**: as specified in XML Events Specification with the restrictions that supported handlers depend on the event. If the handler attribute is not present, the default handler is the parent of the listener element.

The following table summarizes LAsER specific handlers and their associated default action:

| Event | Handler | Default Action |
|----------|------------------------------------|---|
| activate | a | follow the hyperlink |
| activate | conditional | executes the conditional content |
| pause | any element with timing attributes | freezes the time line of this element |
| play | any element with timing attributes | starts or resumes the time line of this element |

The following table summarizes SVG specific handlers and their associated default action, as defined in the SVG specification:

| Event | Handler | Default Action |
|----------|------------------------------------|-----------------------|
| activate | any editable element | starts the editing |
| activate | any element with timing attributes | activates the element |

For other couple (event, handler) not defined in this specification, the default action is undefined.

Replace subclause 6.8.27 "SVG script" with the following:

The **SVG script** element is specified in subclause 18.2 of [W3C SVG11].

Replace subclause 6.8.32 "SVG text" with the following:

6.8.36 SVG text

6.8.36.1 Semantics

The **SVG text** element is defined in subclause 10.4 of [W3C SVG11], with the **SVG text-decoration** attribute from subclause 10.12 of [W3C SVG11].

6.8.36.2 Attributes

- Additional values for the text-decoration attributes are defined, taken from [4]: "LAsER_OUTLINE", "LAsER_EMBOSS", "LAsER_ENGRAVE", "LAsER_LEFTDROPSHADOW", "LAsER_RIGHTDROPSHADOW".

NOTE Authors who want their content to degrade gracefully on pure SVG client should make sure that either: the default value for text-decoration is appropriate or that the requiredExtensions attribute is used.

- SVG font-size: the size of the font represents the height value of the EM-box of a font in the local coordinate system.
- SVG editable: If set to "false" (default) the contents of the text element are not editable in place through the browser. If set to "true", the browser must provide a way for the user to edit the content of the text element and all contained subelements which are not hidden (with visibility="hidden") or disabled (through the switch element or display="none").
- SVG display-align: this attribute is defined in subclause 10.11.5 of [2], i.e. it governs alignment in the direction orthogonal to the main direction of the text.

In subclause 6.8.36.2 "SMIL video Attributes", add this sentence to bullet about clipBegin:

The value of clipBegin is taken into account when the media element is activated.

In subclause 6.8.36.2, add this sentence to bullet about clipEnd:

The value of clipEnd is taken into account when the media element is activated.

In subclause 6.8.36.2, replace the paragraph concerning the semantic of the overlay attribute with the following:

Isr:fullscreen: This attribute can take the values true and false (the default). When set, the video shall be rendered alone and should fill the full screen area. This means no other visual element in the scene shall be rendered and in every other respect, the processing is as normal, e.g. events must still be dispatched to the appropriate elements and audio elements are rendered. The preserveAspectRatio attribute should be respected. If the preserveAspectRatio attribute indicates uniform scaling, then uniform scaling shall be applied. The viewport-fill-opacity attribute should be ignored. The fullscreen attribute is not animatable and not inheritable. This attribute shall not be set on more than one video element at any time in a scene.

Note: The full screen area is the largest part of the whole screen that can be used. The presence of this attribute indicates that the content author wishes to have the video presented as large as possible, and with as few as possible other visual elements (e.g. from other applications or from the OS) on the screen. If the video does not fill the rendering area, the viewport fill color is suggested but not required for the unpainted areas.

In subclause 6.8.37 “Summary of possible attributes per element”, change title of this subclause from “Summary of possible children and attributes per element” to “Summary of possible attributes per element” and renumber to 6.8.41

Replace Table 7 with:

| Element name | Attributes |
|------------------|--|
| a | audio-level color color-rendering display display-align externalResourcesRequired fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight image-rendering line-increment Isr:rotation Isr:scale Isr:translation pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage target text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type |
| animate | accumulate additive attributeName begin by calcMode class dur enabled end fill from id keySplines keyTimes max min repeatCount repeatDur restart to values xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| animateColor | accumulate additive attributeName begin by calcMode class dur enabled end fill from id keySplines keyTimes max min repeatCount repeatDur restart to values xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| animateMotion | accumulate additive attributeName begin by calcMode class dur enabled end fill from id keyPoints keySplines keyTimes max min path repeatCount repeatDur restart rotate to values xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| animateTransform | accumulate additive attributeName begin by calcMode class dur enabled end fill from id keySplines keyTimes max min repeatCount repeatDur restart to type values xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| audio | audio-level begin class dur end externalResourcesRequired id Isr:syncReference repeatCount repeatDur requiredExtensions requiredFeatures requiredFormats syncBehavior syncTolerance systemLanguage type xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space type |
| circle | audio-level class color color-rendering cx cy display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-weight font-variant id image-rendering line-increment Isr:rotation Isr:scale Isr:translation pointer-events r requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |

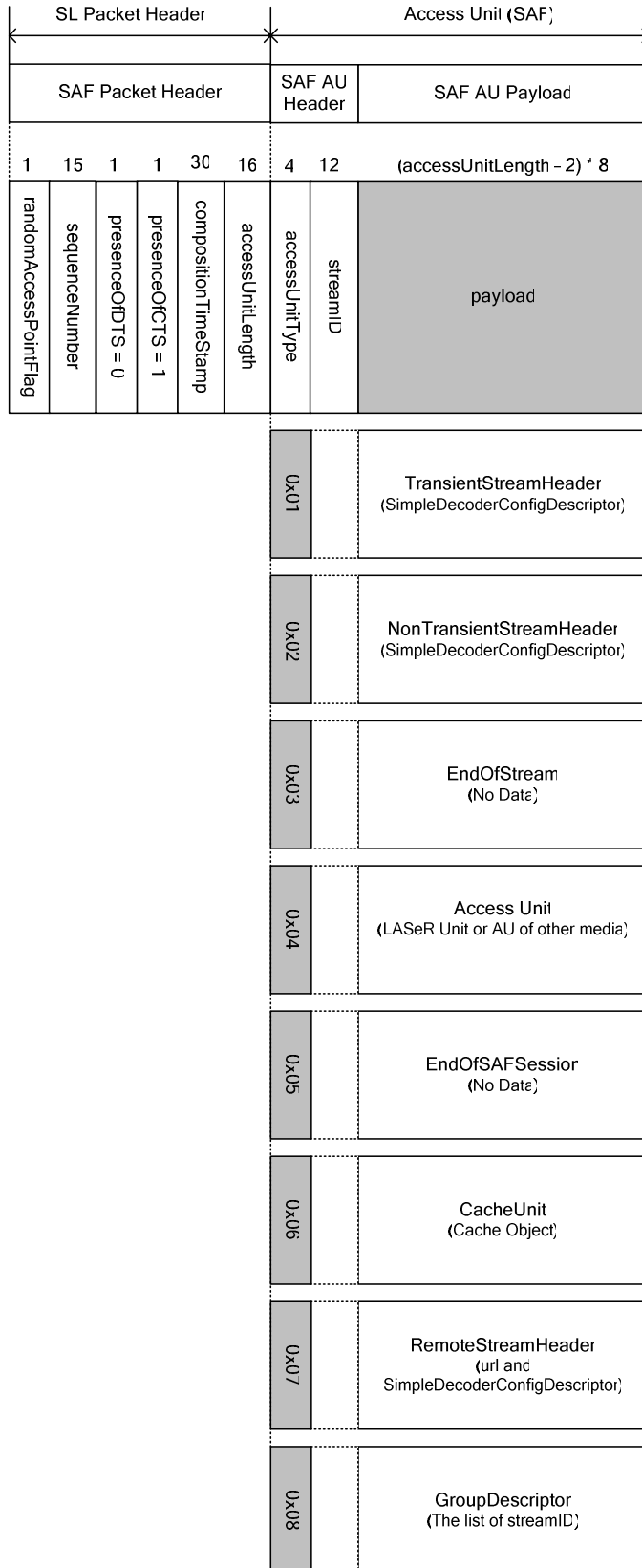
| Element name | Attributes |
|-------------------|---|
| Isr:conditional | id class style begin enabled externalResourcesRequired xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show xml:base xml:lang xml:space |
| Isr:cursorManager | id class style xml:base xml:lang xml:space xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show x y |
| defs | audio-level class color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style font-variant font-weight id image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| desc | class id xml:base xml:lang xml:space |
| ellipse | audio-level class color color-rendering cx cy display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment Isr:rotation Isr:scale Isr:translation pointer-events requiredExtensions requiredFeatures requiredFormats rx ry shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| foreignObject | audio-level class color color-rendering display display-align externalResourcesRequired fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight height id image-rendering line-increment pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility width x xml:base xml:lang xml:space y |
| g | id class style xml:base xml:lang xml:space audio-level color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style text-decoration font-weight font-variant image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering viewport-fill viewport-fill-opacity vector-effect visibility Isr:rotation Isr:scale Isr:translation transform requiredFeatures requiredExtensions systemLanguage requiredFormats requiredFonts nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable focusHighlight externalResourcesRequired |
| image | class display externalResourcesRequired nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable height id Isr:rotation Isr:scale Isr:translation opacity pointer-events requiredExtensions requiredFeatures requiredFormats systemLanguage transform transformBehavior type visibility width x xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space y type preserveAspectRatio viewport-fill viewport-fill-opacity |
| line | audio-level class color color-rendering display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment Isr:rotation Isr:scale Isr:translation pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility x1 x2 xml:base xml:lang xml:space y1 y2 |

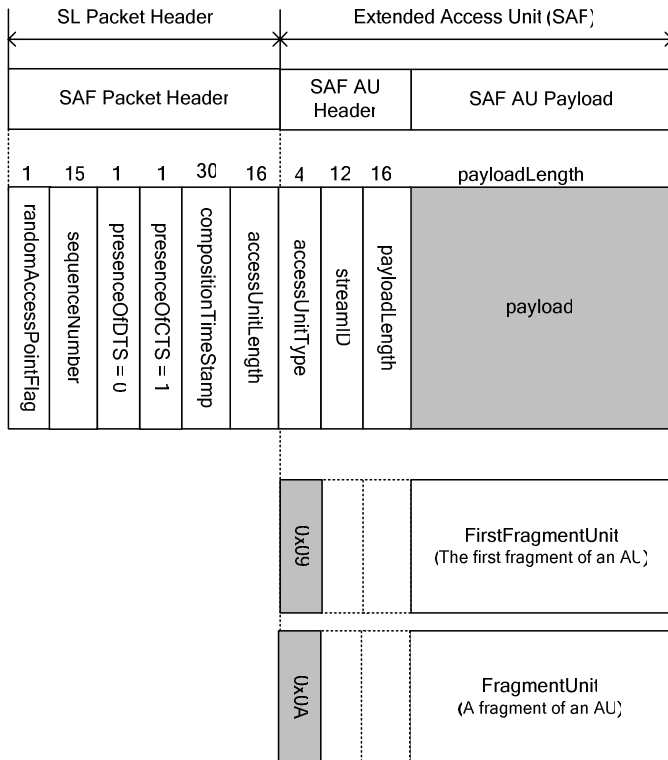
| Element name | Attributes |
|----------------|---|
| linearGradient | audio-level class color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style font-variant font-weight gradient-units id image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility x1 x2 xml:base xml:lang xml:space y1 y2 |
| ev:listener | id enabled event handler observer phase propagate defaultAction target |
| metadata | class id xml:base xml:lang xml:space |
| mpath | class id xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| path | audio-level class color color-rendering d display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment lsr:rotation lsr:scale lsr:translation pathLength pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| polygon | audio-level class color color-rendering display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment lsr:rotation lsr:scale lsr:translation pointer-events points requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| polyline | audio-level class color color-rendering display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment lsr:rotation lsr:scale lsr:translation pointer-events points requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| radialGradient | audio-level class color color-rendering cx cy display display-align fill fill-opacity fill-rule font-family font-size font-style font-variant font-weight gradient-units id image-rendering line-increment pointer-events r shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| rect | audio-level class color color-rendering display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight height id image-rendering line-increment lsr:rotation lsr:scale lsr:translation pointer-events requiredExtensions requiredFeatures requiredFormats rx ry shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility width x xml:base xml:lang xml:space y |

| Element name | Attributes |
|------------------|--|
| lsr:rectClip | id class style xml:base xml:lang xml:space audio-level color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style text-decoration font-weight font-variant image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering viewport-fill viewport-fill-opacity vector-effect visibility lsr:rotation lsr:scale lsr:translation transform requiredFeatures requiredExtensions systemLanguage requiredFormats requiredFonts nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable focusHighlight externalResourcesRequired size |
| script | begin class enabled externalResourcesRequired id type xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space type |
| lsr:selector | id class style xml:base xml:lang xml:space audio-level color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style text-decoration font-weight font-variant image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering viewport-fill viewport-fill-opacity vector-effect visibility lsr:rotation lsr:scale lsr:translation transform requiredFeatures requiredExtensions systemLanguage requiredFormats requiredFonts nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable focusHighlight externalResourcesRequired choice |
| set | attributeName begin class dur enabled end fill id max min repeatCount repeatDur restart to xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space |
| lsr:simpleLayout | id class style xml:base xml:lang xml:space audio-level color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style text-decoration font-weight font-variant image-rendering line-increment pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering viewport-fill viewport-fill-opacity vector-effect visibility lsr:rotation lsr:scale lsr:translation transform requiredFeatures requiredExtensions systemLanguage requiredFormats requiredFonts nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable focusHighlight externalResourcesRequired size |
| stop | audio-level class color color-rendering display display-align fill fill-opacity fill-rule font-family font-size font-style font-variant font-weight id image-rendering line-increment offset pointer-events shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| svg | audio-level baseProfile class color color-rendering contentScriptType display display-align externalResourcesRequired fill fill-opacity fill-rule font-family font-size font-style font-variant font-weight height id image-rendering line-increment playbackOrder pointer-events preserveAspectRatio shape-rendering snapshotTime solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width syncBehaviorDefault syncToleranceDefault text-anchor text-rendering timeLineBegin vector-effect version viewBox viewport-fill viewport-fill-opacity visibility width xml:base xml:lang xml:space zoomAndPan |

| Element name | Attributes |
|--------------|---|
| switch | audio-level class color color-rendering display display-align externalResourcesRequired fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment lsr:rotation lsr:scale lsr:translation pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| text | audio-level color color-rendering display display-align editable fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight image-rendering line-increment lsr:rotation lsr:scale lsr:translation pointer-events requiredExtensions requiredFeatures requiredFormats rotate shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility x y |
| title | class id xml:base xml:lang xml:space |
| tspan | audio-level class color color-rendering display display-align fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility xml:base xml:lang xml:space |
| use | audio-level class color color-rendering display display-align externalResourcesRequired fill fill-opacity fill-rule nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment lsr:rotation lsr:scale lsr:translation overflow pointer-events requiredExtensions requiredFeatures requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width systemLanguage text-anchor text-rendering transform vector-effect viewport-fill viewport-fill-opacity visibility x xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space y |
| video | audio-level begin display dur end externalResourcesRequired fullscreen nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left focusable height lsr:rotation lsr:scale lsr:syncReference lsr:translation overlay pointer-events repeatCount repeatDur requiredExtensions requiredFeatures requiredFormats syncBehavior syncTolerance systemLanguage transform transformBehavior type visibility width x xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type y type preserveAspectRatio viewport-fill viewport-fill-opacity |

In subclause 7.3 "SAF Packet", replace Figure 4, with:





Replace subclause 7.4.1 SAF Packet Header syntax with the following:

```

class SAF_PacketHeader {
    bit(1) randomAccessPointFlag;
    bit(15) sequenceNumber;
    const bit(1) presenceOfDTS = 0;
    const bit(1) presenceOfCTS = 1;
    bit(30) compositionTimeStamp;
    uint(16) accessUnitLength;
}
    
```

Replace subclause 7.4.2 "SAF Packet Header semantics" with the following:

`sequenceNumber` – if present, successive packets shall either have the same sequence number or the value be continuously incremented as a modulo counter. Note – when two packets have the same sequence number, one can be ignored. Sequence numbers are incremented per SAF packet.

`randomAccessPointFlag` – when set to one indicates that random access to the content of this elementary stream is possible here. For SAF packets of type `TransientStreamHeader`, `NonTransientStreamHeader` or `RemoteStreamHeader`, the value of this flag shall be ignored.

`compositionTimeStamp` – is a composition time stamp.

For SAF packets of type `AccessUnit`, the composition time `tc` of the first composition unit resulting from this packet is reconstructed from this composition time stamp according to the formula:

$$tc = (compositionTimeStamp/timeStampResolution + k * 2^{30}/timeStampResolution)$$

where `k` is the number of times that the `compositionTimeStamp` counter has wrapped around. The value of this field from two different packets may be same, if their `streamIDs` are different.

For streams which would require a different decoding time stamp and composition time stamp, such as MPEG-4 Part 10 (AVC) streams, the packets shall be provided in decoding order with composition time stamps, which means that time stamps may not be monotonically growing.

For SAF packets of type `cacheUnit`, this field specifies the validity period in second. In this case, the value 0 is reserved and the value 0x3FFFFFFF means that the Cache Unit never expires.

For SAF packets of type `TransientStreamHeader`, `NonTransientStreamHeader` or `RemoteStreamHeader`, this field specifies when the packet shall be processed.

When `TransientStreamHeader`, `NonTransientStreamHeader` or `RemoteStreamHeader` using a stream ID already in use are received, they shall be ignored by the terminal.

`accessUnitLength` – is the length in bytes of the SAF access unit conveyed in the SAF packet. The value of this field shall be at least 2. Value 1 is reserved for future ISO use. Value 0 signals that the payload of this SAF Packet is a SAF extended AU, as specified in subclause 7.6.

Replace subclause 7.5.2 "SAF Access Unit Semantics" with the following:

7.5.2 Semantics

`accessUnitType` – an indication about the type of the payload. Detailed values of `accessUnitType` and the data corresponding to each type are defined in Table 8

Table 8 — accessUnitType values and corresponding data in the payload

| Value | Type of access unit payload | Data in payload |
|-------------|-----------------------------|--|
| 0x00 | Reserved | - |
| 0x01 | TransientStreamHeader | A SimpleDecoderConfigDescriptor |
| 0x02 | NonTransientStreamHeader | A SimpleDecoderConfigDescriptor |
| 0x03 | EndofStream | (no data) |
| 0x04 | AccessUnit | An Access Unit |
| 0x05 | EndOfSAFSession | (no data) |
| 0x06 | CacheUnit | A cache object |
| 0x07 | RemoteStreamHeader | An url and a SimpleDecoderConfigDescriptor |
| 0x08 ~ 0x0F | Reserved | - |

`streamID` – the reference of the media stream this AU belongs to. This identifier is local to the current SAF session.

`payload` – the data part of the access unit. The size of the payload is signalled by the `accessUnitLength` field in the packet header as specified in 7.4. The type of data in this field is varied by `accessUnitType` as defined in Table 8.

For values of `safAU.accessUnitType` that refer to `TransientStreamHeader` and `NonTransientStreamHeader`, the `payload` shall convey a `SimpleDecoderConfigDescriptor` whose syntax and semantics is specified in 7.6.

A `TransientStreamHeader` shall be used to indicate that the stream shall be used once and by only one media element. Continuous media should be marked as transient.

NOTE Implementations may optimize the resources for a stream using `TransientStreamHeader` as soon as no more media element references the stream.

For values of `safAU.accessUnitType` that refer to LASeR access unit or cache unit, the `payload` shall convey an access unit or cache unit which syntax and semantics are specified in this clause.

For values of `safAU.accessUnitType` that refer to an access unit, the `payload` shall convey an access unit for specific media whose syntax and semantics is opaque to this standard.

In subclause 7.8.2, change the style of "with the addition of an url" to normal.

In subclause 7.9 Cache Unit, replace:

After the end time, the cache object is expired and its SAF content cannot be executed.

with:

After the end time, the cache object is expired and its content cannot be executed.

In subclause 7.9.2 Cache Unit Semantics, remove:

The payload data shall replace the stored presentation referenced by the url of this cache unit when the `safAU.accessUnitType` is 0x06. Otherwise, the payload data shall be appended to the existing presentation.

Create subclause 7.12 "SAF Extended Access Unit":

7.12 SAF Extended Access Unit

7.12.1 Syntax

```
class safXAU {  
    bit(4) accessUnitType;  
    bit(12) streamID;  
    bit(16) payloadLength;  
    byte(8) [payloadLength] payload;  
}
```

7.12.2 Semantics

The value `accessUnitLength` in SAF Packets containing Extended Access Units shall be 0.

`accessUnitType` – the semantic is the same as in subclause 7.5.2.

`streamID` – the semantic is the same as in subclause 7.5.2.

`payload` – the data part of the access unit. The size of the payload is signalled in `payloadLength`. The type of data in this field is varied by `accessUnitType` as defined in Table 8 and in subclause 7.5.2.

Add to subclause 7.11 EndOfSAFSession:

When a SAFSession is closed (either by *EndOfSAFSession*, or by other means), all unused transient media stream originated in this SAF session may be discarded.

We call an unused media stream a stream that has no elements in the scene tree that reference the streamID of this media stream. For efficiency the content of `<sr:conditional>` shall not be taken into account regarding streamID references.

In subclause 8.1 "LAsER Mini Overview", add the following paragraph:

The purpose of LAsER mini is to allow very constrained implementations of LAsER, including J2ME implementations. One specific feature of mini is that the implementation does not need to implement 2 trees, one DOM tree and one composition tree, or otherwise keep the memory of the decoded value for each attribute.

In subclause 8.2.2.5 "LAsER Mini animations", add the following paragraph:

Any content requiring the restoration of a DOM value of an attribute after it has been animated is non conformant. Authors shall make sure that, when deleting an animation or replacing an `xlink:href`, any previously animated value is made irrelevant by any appropriate mean (e.g. deletion of the target element, replacement of the animated value, etc.).

Create subclause 8.2.2.7 "LAsER Mini LAsER Commands"with:

8.2.2.7 LAsER Commands

color-rendering, shape-rendering, text-rendering, image-rendering are not updatable in this profile.

In subclause 8.2.4 « Profile Definition », replace table with:

| Element name | Attributes |
|------------------|--|
| a | id externalResourcesRequired display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform xlink:href |
| animate | id Isr:enabled begin end dur repeatCount repeatDur restart attributeName fill="freeze" to xlink:href(no replace) from by values calcMode keyTimes keySplines additive="replace" accumulate |
| animateColor | id Isr:enabled begin end dur repeatCount repeatDur restart attributeName fill="freeze" to xlink:href(no replace) from by values calcMode keyTimes keySplines additive="replace" accumulate |
| animateMotion | id path keyPoints rotate Isr:enabled begin end dur repeatCount repeatDur restart fill="freeze" to xlink:href(no replace) from by values calcMode keyTimes keySplines additive="replace" accumulate |
| animateTransform | id type Isr:enabled begin end dur repeatCount repeatDur restart attributeName fill="freeze" to xlink:href(no replace) from by values calcMode keyTimes keySplines additive="replace" accumulate |
| audio | id externalResourcesRequired begin end clipBegin clipEnd repeatCount repeatDur syncBehavior syncTolerance audio-level syncReference requiredFeatures requiredExtensions systemLanguage requiredFormats xlink:href |
| circle | id cx cy r display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| conditional | externalResourcesRequired id begin type Isr:enabled |
| cursorManager | id x y xlink:href |
| defs | id |
| desc | id |
| ellipse | id rx ry cx cy display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| foreignObject | id externalResourcesRequired width height x y viewport-fill viewport-fill-opacity display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats |
| g | id externalResourcesRequired display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| image | id externalResourcesRequired width height x y transformBehavior(no geometric) opacity display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform xlink:href |
| line | id x1 y1 x2 y2 display stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| linearGradient | id gradient-units x1 x2 y1 y2 display visibility |
| ev:listener | id Isr:enabled delay event observer timeAttribute handler propagate defaultAction target |
| metadata | id |
| mpath | id xlink:href |
| path | id d display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| polygon | id points display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| polyline | id points display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| radialGradient | id gradient-units cx cy r display visibility |
| rect | id width height x y rx ry display fill fill-opacity fill-rule stroke stroke-opacity stroke-width visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| rectClip | id externalResourcesRequired size display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| selector | id externalResourcesRequired choice display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| set | id Isr:enabled begin end dur repeatCount repeatDur restart attributeName fill="freeze" to xlink:href(no replace) |
| simpleLayout | id externalResourcesRequired delta display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform |
| stop | id offset display stop-color stop-opacity visibility |
| svg | id externalResourcesRequired width(not animatable if gradient with gradientUnits="userSpaceOnUse") height(not animatable if gradient with gradientUnits="userSpaceOnUse") viewBox(not animatable if gradient with gradientUnits="userSpaceOnUse") preserveAspectRatio zoomAndPan viewport-fill viewport-fill-opacity syncBehavior syncBehaviorDefault syncTolerance syncToleranceDefault display pointer-events visibility color-rendering shape-rendering image-rendering text-rendering |
| switch | id externalResourcesRequired display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats |
| text | id display display-align fill fill-opacity fill-rule font-family font-size font-style font-weight line-increment stroke stroke-opacity stroke-width text-anchor visibility requiredFeatures requiredExtensions systemLanguage requiredFormats editable x y transform text-decoration |
| title | id |

| | |
|-------|---|
| tspan | id display fill fill-opacity fill-rule font-family font-size font-style font-weight line-increment stroke stroke-opacity stroke-width text-anchor visibility requiredFeatures requiredExtensions systemLanguage requiredFormats text-decoration |
| use | id externalResourcesRequired overflow x y display visibility requiredFeatures requiredExtensions systemLanguage requiredFormats transform xlink:href |
| video | id externalResourcesRequired begin end clipBegin clipEnd repeatCount repeatDur syncBehavior syncBehaviorDefault syncTolerance syncToleranceDefault audio-level syncReference requiredFeatures requiredExtensions systemLanguage requiredFormats xlink:href width height x y overlay="top LAsER_fullscreen" transformBehavior(no geometric) transform |

In subclause 12.1.4 "Animations and updates", replace Table 11 with the following:

| Type | Encoded value |
|-------------------|---------------|
| ANIMTYPE_string | 0 |
| ANIMTYPE_float | 1 |
| ANIMTYPE_path | 2 |
| ANIMTYPE_pointSeq | 3 |
| ANIMTYPE_fraction | 4 |
| ANIMTYPE_color | 5 |
| ANIMTYPE_enum | 6 |
| reserved | 7 |
| ANIMTYPE_floats | 8 |
| ANIMTYPE_point | 9 |
| ANIMTYPE_id | 10 |
| ANIMTYPE_font | 11 |
| ANIMTYPE_uri | 12 |

Replace Table 12 with the following:

| Attribute name | Encoded type for animation | Attribute name | Encoded type for animation |
|-----------------|----------------------------|-----------------------|----------------------------|
| a.target | ANIMTYPE_string | r | ANIMTYPE_float |
| audio-level | ANIMTYPE_fraction | rotate | ANIMTYPE_float |
| choice | ANIMTYPE_enum | rx | ANIMTYPE_float |
| color | ANIMTYPE_color | ry | ANIMTYPE_float |
| color-rendering | ANIMTYPE_enum | shape-rendering | ANIMTYPE_enum |
| cx | ANIMTYPE_float | size | ANIMTYPE_point |
| cy | ANIMTYPE_float | stop-color | ANIMTYPE_color |
| d | ANIMTYPE_path | stop-opacity | ANIMTYPE_fraction |
| display | ANIMTYPE_enum | stroke | ANIMTYPE_color |
| display-align | ANIMTYPE_enum | stroke-dasharray | ANIMTYPE_floats |
| editable | ANIMTYPE_enum | stroke-dashoffset | ANIMTYPE_float |
| fill | ANIMTYPE_color | stroke-linecap | ANIMTYPE_enum |
| fill-opacity | ANIMTYPE_fraction | stroke-linejoin | ANIMTYPE_enum |
| fill-rule | ANIMTYPE_enum | stroke-miterlimit | ANIMTYPE_float |
| nav-next | ANIMTYPE_id | stroke-opacity | ANIMTYPE_fraction |
| nav-prev | ANIMTYPE_id | stroke-width | ANIMTYPE_float |
| nav-right | ANIMTYPE_id | target | ANIMTYPE_string |
| nav-up | ANIMTYPE_id | text-anchor | ANIMTYPE_enum |
| nav-up-right | ANIMTYPE_id | transform | ANIMTYPE_floats |
| nav-up-left | ANIMTYPE_id | type | ANIMTYPE_enum |
| nav-down | ANIMTYPE_id | vector-effect | ANIMTYPE_enum |
| nav-down-right | ANIMTYPE_id | viewBox | ANIMTYPE_floats |
| nav-down-left | ANIMTYPE_id | viewport-fill | ANIMTYPE_color |
| nav-left | ANIMTYPE_id | viewport-fill-opacity | ANIMTYPE_fraction |
| focusable | ANIMTYPE_id | visibility | ANIMTYPE_enum |
| font-family | ANIMTYPE_font | width | ANIMTYPE_float |
| font-size | ANIMTYPE_float | x | ANIMTYPE_float |
| font-style | ANIMTYPE_enum | x1 | ANIMTYPE_float |

| | | | |
|-----------------|---------------------|-------------------|----------------|
| font-weight | ANIMTYPE_enum | x2 | ANIMTYPE_float |
| gradientUnits | ANIMTYPE_enum | xlink:href | ANIMTYPE_uri |
| height | ANIMTYPE_float | y | ANIMTYPE_float |
| image-rendering | ANIMTYPE_enum | y1 | ANIMTYPE_float |
| listener.target | ANIMTYPE_id | y2 | ANIMTYPE_float |
| | opacity | ANIMTYPE_fraction | |
| | pathLength | ANIMTYPE_float | |
| | pointer-events | ANIMTYPE_ints | |
| | points | ANIMTYPE_pointSeq | |
| | preserveAspectRatio | ANIMTYPE_enum | |

Replace subclause 12.1.5 "Encoding of enumerations" with the following:

The SDL described in subclause 12.2 makes use of the following syntax: "// Enumeration:" followed by a space separated list of "string{value}", where *value* is the decimal value used to encode the *string* in the bit field following on the next line. Values not listed but encodable in this bit field are ISO reserved.

In subclause 12.2 "Binary Syntax", replace the content of subclause 12.2 by the following:

12.2.1 Main Structure

```

class element_a {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_target;
    if(has_target) {
        attr_custom_byteAlignedString target;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class privateAttributeContainer {
    bit(4) ch4;
    switch(ch4){
        case 0:
            privateAnyXMLAttribute anyXML;
            break;
        case 1:
            privateOpaqueAttribute opaque;
            break;
        case 2:
            attr_any reserved;
            break;
        default:
            attr_custom_extension ext;
            break;
    }
}
class elements {
    bit(6) ch4;
    switch(ch4){
        case 0:
            element_a a;
            break;
        case 1:
            element_animate animate;

```

```

    break;
case 2:
    element_animate animateColor;
    break;
case 3:
    element_animateMotion animateMotion;
    break;
case 4:
    element_animateTransform animateTransform;
    break;
case 5:
    element_audio audio;
    break;
case 6:
    element_circle circle;
    break;
case 7:
    element_defs defs;
    break;
case 8:
    element_desc_metadata_title desc;
    break;
case 9:
    element_ellipse ellipse;
    break;
case 10:
    element_foreignObject foreignObject;
    break;
case 11:
    element_g g;
    break;
case 12:
    element_image image;
    break;
case 13:
    element_line line;
    break;
case 14:
    element_linearGradient linearGradient;
    break;
case 15:
    element_desc_metadata_title metadata;
    break;
case 16:
    element_mpath mpath;
    break;
case 17:
    element_path path;
    break;
case 18:
    element_polygon polygon;
    break;
case 19:
    element_polygon polyline;
    break;
case 20:
    element_radialGradient radialGradient;
    break;
case 21:
    element_rect rect;
    break;
case 22:
    element_sameg sameg;
    break;
case 23:
    element_sameline sameline;
    break;
case 24:
    element_samepath samepath;
    break;
case 25:
    element_samepathfill samepathfill;
    break;
case 26:
    element_samepolygon samepolygon;
    break;
case 27:
    element_samepolygonfill samepolygonfill;
    break;
case 28:
    element_samepolygonstroke samepolygonstroke;
    break;
case 29:
    element_samepolygon samepolyline;
    break;
case 30:
    element_samepolygonfill samepolylinefill;
    break;
case 31:
    element_samepolygonstroke samepolylinestroke;

```

```

        break;
    case 32:
        element_samerect samerect;
        break;
    case 33:
        element_samerectfill samerectfill;
        break;
    case 34:
        element_sametext sametext;
        break;
    case 35:
        element_sametextfill sametextfill;
        break;
    case 36:
        element_sameuse sameuse;
        break;
    case 37:
        element_script script;
        break;
    case 38:
        element_set set;
        break;
    case 39:
        element_stop stop;
        break;
    case 40:
        element_switch switch;
        break;
    case 41:
        element_text text;
        break;
    case 42:
        element_desc_metaData_title title;
        break;
    case 43:
        element_tspan tspan;
        break;
    case 44:
        element_use use;
        break;
    case 45:
        element_video video;
        break;
    case 46:
        element_listener listener;
        break;
    case 47:
        element_conditional conditional;
        break;
    case 48:
        element_cursorManager cursorManager;
        break;
    case 49:
        element_any extElement;
        break;
    case 50:
        privateElementContainer privateElementContainer;
        break;
    case 51:
        element_rectClip rectClip;
        break;
    case 52:
        element_selector selector;
        break;
    case 53:
        element_simpleLayout simpleLayout;
        break;
    case 54:
        attr_custom_byteAlignedString textContent;
        break;
    }
}
class element_animate {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_attributeName;
    if(has_attributeName) {
        attr_AttributeName attributeName;
    }
    bit(1) has_accumulate;
    if(has_accumulate) {
        attr_accumulate accumulate;
    }
}
bit(1) has_additive;

```

```

if(has_additive) {
    attr_additive additive;
}
bit(1) has_by;
if(has_by) {
    attr_custom_AnimatedValue by;
}
bit(1) has_calcMode;
if(has_calcMode) {
    attr_calcMode calcMode;
}
bit(1) has_from;
if(has_from) {
    attr_custom_AnimatedValue from;
}
bit(1) has_keySplines;
if(has_keySplines) {
    attr_custom_fraction12List keySplines;
}
bit(1) has_keyTimes;
if(has_keyTimes) {
    attr_custom_fraction12List keyTimes;
}
bit(1) has_values;
if(has_values) {
    attr_custom_AnimatedValues values;
}
bit(1) has_attributeType;
if(has_attributeType) {
    attr_attributeType attributeType;
}
bit(1) has_begin;
if(has_begin) {
    attr_smil_times begin;
}
bit(1) has_dur;
if(has_dur) {
    attr_time dur;
}
bit(1) has_fill;
if(has_fill) {
    attr_animFill fill;
}
bit(1) has_repeatCount;
if(has_repeatCount) {
    attr_repeatCount repeatCount;
}
bit(1) has_repeatDur;
if(has_repeatDur) {
    attr_repeatDur repeatDur;
}
bit(1) has_restart;
if(has_restart) {
    attr_restart restart;
}
bit(1) has_to;
if(has_to) {
    attr_custom_AnimatedValue to;
}
bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
bit(1) enabled;
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_animateMotion {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_accumulate;
    if(has_accumulate) {
        attr_accumulate accumulate;
    }
    bit(1) has_additive;
    if(has_additive) {
        attr_additive additive;
    }
    bit(1) has_by;
    if(has_by) {
        attr_custom_AnimatedValue by;
    }
}

```



```

}
bit(1) has_calcMode;
if(has_calcMode) {
    attr_calcMode calcMode;
}
bit(1) has_from;
if(has_from) {
    attr_custom_AnimatedValue from;
}
bit(1) has_keySplines;
if(has_keySplines) {
    attr_custom_fraction12List keySplines;
}
bit(1) has_keyTimes;
if(has_keyTimes) {
    attr_custom_fraction12List keyTimes;
}
bit(1) has_values;
if(has_values) {
    attr_custom_AnimatedValues values;
}
bit(1) has_attributeType;
if(has_attributeType) {
    attr_attributeType attributeType;
}
bit(1) has_begin;
if(has_begin) {
    attr_smil_times begin;
}
bit(1) has_dur;
if(has_dur) {
    attr_time dur;
}
bit(1) has_fill;
if(has_fill) {
    attr_animFill fill;
}
bit(1) has_repeatCount;
if(has_repeatCount) {
    attr_repeatCount repeatCount;
}
bit(1) has_repeatDur;
if(has_repeatDur) {
    attr_repeatDur repeatDur;
}
bit(1) has_restart;
if(has_restart) {
    attr_restart restart;
}
bit(1) has_to;
if(has_to) {
    attr_custom_AnimatedValue to;
}
bit(1) has_keyPoints;
if(has_keyPoints) {
    attr_floatList keyPoints;
}
bit(1) has_path;
if(has_path) {
    attr_custom_path path;
}
bit(1) has_rotate;
if(has_rotate) {
    attr_rotate rotate;
}
bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
bit(1) enabled;
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_animateTransform {
bit(1) has_id;
if(has_id) {
    attr_custom_ID id;
}
bit(1) has_rare;
if(has_rare) {
    attr_custom_rare rare;
}
bit(1) has_attributeName;
if(has_attributeName) {
    attr_AttributeName attributeName;
}
}
attr_rotscatra type;

```

```

bit(1) has_accumulate;
if(has_accumulate) {
    attr_accumulate accumulate;
}
bit(1) has_additive;
if(has_additive) {
    attr_additive additive;
}
bit(1) has_by;
if(has_by) {
    attr_custom_AnimatedValue by;
}
bit(1) has_calcMode;
if(has_calcMode) {
    attr_calcMode calcMode;
}
bit(1) has_from;
if(has_from) {
    attr_custom_AnimatedValue from;
}
bit(1) has_keySplines;
if(has_keySplines) {
    attr_custom_fraction12List keySplines;
}
bit(1) has_keyTimes;
if(has_keyTimes) {
    attr_custom_fraction12List keyTimes;
}
bit(1) has_values;
if(has_values) {
    attr_custom_AnimatedValues values;
}
bit(1) has_attributeType;
if(has_attributeType) {
    attr_attributeType attributeType;
}
bit(1) has_begin;
if(has_begin) {
    attr_smil_times begin;
}
bit(1) has_dur;
if(has_dur) {
    attr_time dur;
}
bit(1) has_fill;
if(has_fill) {
    attr_animFill fill;
}
bit(1) has_repeatCount;
if(has_repeatCount) {
    attr_repeatCount repeatCount;
}
bit(1) has_repeatDur;
if(has_repeatDur) {
    attr_repeatDur repeatDur;
}
bit(1) has_restart;
if(has_restart) {
    attr_restart restart;
}
bit(1) has_to;
if(has_to) {
    attr_custom_AnimatedValue to;
}
bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
bit(1) enabled;
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_audio {
bit(1) has_id;
if(has_id) {
    attr_custom_ID id;
}
bit(1) has_rare;
if(has_rare) {
    attr_custom_rare rare;
}
bit(1) has_begin;
if(has_begin) {
    attr_smil_times begin;
}
bit(1) has_dur;
if(has_dur) {

```

```

    attr_time dur;
}
bit(1) externalResourcesRequired;
bit(1) has_repeatCount;
if(has_repeatCount) {
    attr_repeatCount repeatCount;
}
bit(1) has_repeatDur;
if(has_repeatDur) {
    attr_repeatDur repeatDur;
}
bit(1) has_restart;
if (has_restart) {
    attr_restart restart;
}
}
bit(1) has_syncBehavior;
if(has_syncBehavior) {
    attr_syncBehavior syncBehavior;
}
bit(1) has_syncTolerance;
if(has_syncTolerance) {
    attr_syncTolerance syncTolerance;
}
}
bit(1) has_type;
if(has_type) {
    attr_custom_byteAlignedString type;
}
bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
}
bit(1) has_clipBegin;
if(has_clipBegin) {
    attr_time clipBegin;
}
}
bit(1) has_clipEnd;
if(has_clipEnd) {
    attr_time clipEnd;
}
}
bit(1) has_syncReference;
if(has_syncReference) {
    attr_custom_anyURI syncReference;
}
}
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
}
object_content child0;
}
}
class element_circle {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
}
bit(1) has_rare;
if(has_rare) {
    attr_custom_rare rare;
}
}
bit(1) has_fill;
if(has_fill) {
    attr_custom_paint fill;
}
}
bit(1) has_stroke;
if(has_stroke) {
    attr_custom_paint stroke;
}
}
bit(1) has_cx;
if(has_cx) {
    attr_custom_coordinate cx;
}
}
bit(1) has_cy;
if(has_cy) {
    attr_custom_coordinate cy;
}
}
attr_custom_coordinate r;
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
}
object_content child0;
}
}
class element_defs {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
}
bit(1) has_rare;
if(has_rare) {
    attr_custom_rare rare;
}
}
}

```

```

    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_desc_metaData_title {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_ellipse {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_cx;
    if(has_cx) {
        attr_custom_coordinate cx;
    }
    bit(1) has_cy;
    if(has_cy) {
        attr_custom_coordinate cy;
    }
    attr_custom_coordinate rx;
    attr_custom_coordinate ry;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_foreignObject {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) externalResourcesRequired;
    attr_custom_coordinate height;
    attr_custom_coordinate width;
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
}

```

```

bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
bit(1) opt_group;
if(opt_group) {
    vluimsbf5 occl;
    for(int t=0;t<occl;t++) {
        privateElementContainer child0[[t]];
    }
}
}
class element_g {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_image {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_height;
    if(has_height) {
        attr_custom_coordinate height;
    }
    bit(1) has_opacity;
    if(has_opacity) {
        attr_custom_0tofloat opacity;
    }
    bit(1) has_preserveAspectRatio;
    if(has_preserveAspectRatio) {
        attr_preserveAspectRatio preserveAspectRatio;
    }
    bit(1) has_type;
    if(has_type) {
        attr_custom_byteAlignedString type;
    }
    bit(1) has_width;
    if(has_width) {
        attr_custom_coordinate width;
    }
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    bit(1) has_transformBehavior;
    if(has_transformBehavior) {
        attr_transformBehavior transformBehavior;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}

```

```

    object_content child0;
}
class element_line {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_x1;
    if(has_x1) {
        attr_custom_coordinate x1;
    }
    attr_custom_coordinate x2;
    bit(1) has_y1;
    if(has_y1) {
        attr_custom_coordinate y1;
    }
    attr_custom_coordinate y2;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_linearGradient {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_gradientUnits;
    if(has_gradientUnits) {
        attr_gradientUnits gradientUnits;
    }
    bit(1) has_x1;
    if(has_x1) {
        attr_custom_coordinate x1;
    }
    bit(1) has_x2;
    if(has_x2) {
        attr_custom_coordinate x2;
    }
    bit(1) has_y1;
    if(has_y1) {
        attr_custom_coordinate y1;
    }
    bit(1) has_y2;
    if(has_y2) {
        attr_custom_coordinate y2;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_mpath {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
}

```

```

    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_path {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_custom_path d;
    bit(1) has_pathLength;
    if(has_pathLength) {
        attr_custom_fixed_16_8 pathLength;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_polygon {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_custom_pointSequence points;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_radialGradient {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_cx;
    if(has_cx) {
        attr_custom_coordinate cx;
    }
    bit(1) has_cy;
    if(has_cy) {
        attr_custom_coordinate cy;
    }
    bit(1) has_gradientUnits;
    if(has_gradientUnits) {
        attr_gradientUnits gradientUnits;
    }
    bit(1) has_r;
    if(has_r) {

```

```

    attr_custom_coordinate r;
}
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_rect {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_custom_coordinate height;
    bit(1) has_rx;
    if(has_rx) {
        attr_custom_coordinate rx;
    }
    bit(1) has_ry;
    if(has_ry) {
        attr_custom_coordinate ry;
    }
    attr_custom_coordinate width;
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_sameg {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    objectSame_content child0;
}
class element_sameline {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_x1;
    if(has_x1) {
        attr_custom_coordinate x1;
    }
    attr_custom_coordinate x2;
    bit(1) has_y1;
    if(has_y1) {
        attr_custom_coordinate y1;
    }
    attr_custom_coordinate y2;
    objectSame_content child0;
}
class element_samepath {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    attr_custom_path d;
    objectSame_content child0;
}
class element_samepathfill {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
}

```



```

    attr_custom_path d;
    objectSame_content child0;
}
class element_samepolygon {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    attr_custom_pointSequence points;
    objectSame_content child0;
}
class element_samepolygonfill {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    attr_custom_pointSequence points;
    objectSame_content child0;
}
class element_samepolygonstroke {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_custom_pointSequence points;
    objectSame_content child0;
}
class element_samerect {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    attr_custom_coordinate height;
    attr_custom_coordinate width;
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
    objectSame_content child0;
}
class element_samerectfill {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    attr_custom_coordinate height;
    attr_custom_coordinate width;
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
    objectSame_content child0;
}
class element_sametext {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_x;
    if(has_x) {
        attr_coordinateList x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_coordinateList y;
    }
    objectSame_content child0;
}
class element_sametextfill {
    bit(1) has_id;

```

```

    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_x;
    if(has_x) {
        attr_coordinateList x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_coordinateList y;
    }
    objectSame_content child0;
}
class element_sameuse {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    objectSame_content child0;
}
class element_script {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_type;
    if(has_type) {
        attr_script type;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_set {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_attributeName;
    if(has_attributeName) {
        attr_AttributeName attributeName;
    }
    bit(1) has_attributeType;
    if(has_attributeType) {
        attr_attributeType attributeType;
    }
    bit(1) has_begin;
    if(has_begin) {
        attr_smil_times begin;
    }
    bit(1) has_dur;
    if(has_dur) {
        attr_time dur;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_animFill fill;
    }
    bit(1) has_repeatCount;
    if(has_repeatCount) {
        attr_repeatCount repeatCount;
    }
    bit(1) has_repeatDur;
    if(has_repeatDur) {
        attr_repeatDur repeatDur;
    }
    bit(1) has_restart;

```

```

    if(has_restart) {
        attr_restart restart;
    }
    bit(1) has_to;
    if(has_to) {
        attr_custom_AnimatedValue to;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    bit(1) enabled;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_stop {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_custom_fixed_16_8 offset;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_switch {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_text {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    attr_editable editable;
    bit(1) has_rotate;
    if(has_rotate) {
        attr_floatList rotate;
    }
    bit(1) has_x;
    if(has_x) {

```

```

    attr_coordinateList x;
  }
  bit(1) has_y;
  if(has_y) {
    attr_coordinateList y;
  }
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_tspan {
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if(has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_fill;
  if(has_fill) {
    attr_custom_paint fill;
  }
  bit(1) has_stroke;
  if(has_stroke) {
    attr_custom_paint stroke;
  }
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_use {
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if(has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_fill;
  if(has_fill) {
    attr_custom_paint fill;
  }
  bit(1) has_stroke;
  if(has_stroke) {
    attr_custom_paint stroke;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_overflow;
  if(has_overflow) {
    attr_overflow overflow;
  }
  bit(1) has_x;
  if(has_x) {
    attr_custom_coordinate x;
  }
  bit(1) has_y;
  if(has_y) {
    attr_custom_coordinate y;
  }
  bit(1) has_href;
  if(has_href) {
    attr_custom_anyURI href;
  }
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_video {
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if(has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_begin;
  if(has_begin) {
    attr_smil_times begin;
  }
  bit(1) has_dur;
  if(has_dur) {

```

```

    attr_time dur;
}
bit(1) externalResourcesRequired;
bit(1) has_height;
if(has_height) {
    attr_custom_coordinate height;
}
bit(1) has_overlay;
if(has_overlay) {
    attr_overlay overlay;
}
bit(1) has_preserveAspectRatio;
if(has_preserveAspectRatio) {
    attr_preserveAspectRatio preserveAspectRatio;
}
bit(1) has_repeatCount;
if(has_repeatCount) {
    attr_repeatCount repeatCount;
}
bit(1) has_repeatDur;
if(has_repeatDur) {
    attr_repeatDur repeatDur;
}
bit(1) has_restart;
if (has_restart) {
    attr_restart restart;
}
bit(1) has_syncBehavior;
if(has_syncBehavior) {
    attr_syncBehavior syncBehavior;
}
bit(1) has_syncTolerance;
if(has_syncTolerance) {
    attr_syncTolerance syncTolerance;
}
bit(1) has_transformBehavior;
if(has_transformBehavior) {
    attr_transformBehavior transformBehavior;
}
bit(1) has_type;
if(has_type) {
    attr_custom_byteAlignedString type;
}
bit(1) has_width;
if(has_width) {
    attr_custom_coordinate width;
}
bit(1) has_x;
if(has_x) {
    attr_custom_coordinate x;
}
bit(1) has_y;
if(has_y) {
    attr_custom_coordinate y;
}
bit(1) has_href;
if(has_href) {
    attr_custom_anyURI href;
}
bit(1) has_clipBegin;
if(has_clipBegin) {
    attr_time clipBegin;
}
bit(1) has_clipEnd;
if(has_clipEnd) {
    attr_time clipEnd;
}
bit(1) has_fullscreen;
if (has_fullscreen) {
    bit(1) fullscreen;
}
bit(1) has_syncReference;
if(has_syncReference) {
    attr_custom_anyURI syncReference;
}
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_listener {
bit(1) has_id;
if(has_id) {
    attr_custom_ID id;
}
bit(1) has_rare;
if(has_rare) {
    attr_custom_rare rare;
}
}

```

```

bit(1) has_defaultAction;
if(has_defaultAction) {
    attr_defaultAction defaultAction;
}
bit(1) has_event;
if(has_event) {
    attr_custom_event event;
}
bit(1) has_handler;
if(has_handler) {
    attr_custom_anyURI handler;
}
bit(1) has_observer;
if(has_observer) {
    attr_custom_IDREF observer;
}
attr_phase phase;
bit(1) has_propagate;
if(has_propagate) {
    attr_propagate propagate;
}
bit(1) has_target;
if(has_target) {
    attr_custom_IDREF target;
}
bit(1) enabled;
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
object_content child0;
}
class element_conditional {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_begin;
    if(has_begin) {
        attr_smil_times begin;
    }
    bit(1) externalResourcesRequired;
    bit(1) enabled;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    updateListType_content child0;
    bit(1) opt_group;
    if(opt_group) {
        privateAttributeContainer privateAttributes;
    }
}
class updateListType_content {
    vluimsbf5 encoding-length;
    attr_custom_align encoding-align-before;
    vluimsbf5 occ0;
    for(int t=0;t<occ0+1;t++) {
        updates child0;
    }
    attr_custom_align encoding-align-after;
}
class updates {
    bit(4) ch4;
    switch(ch4){
        case 0:
            update_Add Add;
            break;
        case 1:
            update_Clean Clean;
            break;
        case 2:
            update_Delete Delete;
            break;
        case 3:
            update_Insert Insert;
            break;
        case 4:
            update_NewScene NewScene;
            break;
        case 5:
            update_RefreshScene RefreshScene;
            break;
        case 6:
            update_Replace Replace;
            break;
    }
}

```

```

    case 7:
        update_Restore Restore;
        break;
    case 8:
        update_Save Save;
        break;
    case 9:
        update_SendEvent SendEvent;
        break;
    case 10:
        element_any ext;
        break;
    case 11:
        attr_custom_byteAlignedString textContent;
        break;
    default:
        break;
    }
}
class update_Add {
    bit(1) has_attributeName;
    if(has_attributeName) {
        attr_AttributeName attributeName;
    }
    bit(1) has_operandAttribute;
    if(has_operandAttribute) {
        attr_AttributeName operandAttribute;
    }
    bit(1) has_operandElementId;
    if(has_operandElementId) {
        attr_custom_IDREF operandElementId;
    }
    attr_custom_IDREF ref;
    bit(1) has_value;
    if(has_value) {
        attr_custom_updateValue value;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
class update_Clean {
    attr_custom_byteAlignedString groupID;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
class update_Delete {
    bit(1) has_attributeName;
    if(has_attributeName) {
        attr_AttributeName attributeName;
    }
    bit(1) has_index;
    if(has_index) {
        attr_index index;
    }
    attr_custom_IDREF ref;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
class update_Insert {
    bit(1) has_attributeName;
    if(has_attributeName) {
        attr_AttributeName attributeName;
    }
    bit(1) has_index;
    if(has_index) {
        attr_index index;
    }
    attr_custom_IDREF ref;
    bit(1) has_value;
    if(has_value) {
        attr_custom_updateValue value;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    bit(1) opt_group;
    if(opt_group) {
        updatable_elements child0;
    }
}
class updatable_elements {
    bit(1) ch4;
    switch(ch4){

```

```

case 0:
  bit(6) ch6;
  switch(ch6){
    case 0:
      element_a a;
      break;
    case 1:
      element_animate animate;
      break;
    case 2:
      element_animate animateColor;
      break;
    case 3:
      element_animateMotion animateMotion;
      break;
    case 4:
      element_animateTransform animateTransform;
      break;
    case 5:
      element_audio audio;
      break;
    case 6:
      element_circle circle;
      break;
    case 7:
      element_defs defs;
      break;
    case 8:
      element_desc_metadata_title desc;
      break;
    case 9:
      element_ellipse ellipse;
      break;
    case 10:
      element_foreignObject foreignObject;
      break;
    case 11:
      element_g g;
      break;
    case 12:
      element_image image;
      break;
    case 13:
      element_line line;
      break;
    case 14:
      element_linearGradient linearGradient;
      break;
    case 15:
      element_desc_metadata_title metadata;
      break;
    case 16:
      element_mpath mpath;
      break;
    case 17:
      element_path path;
      break;
    case 18:
      element_polygon polygon;
      break;
    case 19:
      element_polygon polyline;
      break;
    case 20:
      element_radialGradient radialGradient;
      break;
    case 21:
      element_rect rect;
      break;
    case 22:
      element_script script;
      break;
    case 23:
      element_set set;
      break;
    case 24:
      element_stop stop;
      break;
    case 25:
      element_svg svg;
      break;
    case 26:
      element_switch switch;
      break;
    case 27:
      element_text text;
      break;
    case 28:
      element_desc_metadata_title title;
      break;
  }

```



```

    case 29:
        element_tspan tspan;
        break;
    case 30:
        element_use use;
        break;
    case 31:
        element_video video;
        break;
    case 32:
        element_listener listener;
        break;
    }
    break;
case 1:
    bit(3) ch61;
    switch(ch61){
        case 0:
            element_conditional conditional;
            break;
        case 1:
            element_cursorManager cursorManager;
            break;
        case 2:
            element_any extElement;
            break;
        case 3:
            privateElementContainer privateElement;
            break;
        case 4:
            element_rectClip rectClip;
            break;
        case 5:
            element_selector selector;
            break;
        case 6:
            element_simpleLayout simpleLayout;
            break;
    }
    break;
}
}
class element_svg {
    bit(1) has_id;
    if(has_id){
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) has_baseProfile;
    if(has_baseProfile) {
        attr_custom_byteAlignedString baseProfile;
    }
    bit(1) has_contentScriptType;
    if(has_contentScriptType) {
        attr_custom_byteAlignedString contentScriptType;
    }
    bit(1) externalResourcesRequired;
    attr_custom_valueWithUnits height;
    bit(1) has_playbackOrder;
    if(has_playbackOrder) {
        attr_playbackOrder playbackOrder;
    }
    bit(1) has_preserveAspectRatio;
    if(has_preserveAspectRatio) {
        attr_preserveAspectRatio preserveAspectRatio;
    }
    bit(1) has_snapshotTime;
    if(has_snapshotTime) {
        attr_time snapshotTime;
    }
    bit(1) has_syncBehaviorDefault;
    if(has_syncBehaviorDefault) {
        attr_syncBehaviorDefault syncBehaviorDefault;
    }
    bit(1) has_syncToleranceDefault;
    if(has_syncToleranceDefault) {
        attr_syncToleranceDefault syncToleranceDefault;
    }
    bit(1) has_timelineBegin;

```

```

    if(has_timelineBegin) {
        attr_timeLineBegin timelineBegin;
    }
    bit(1) has_version;
    if(has_version) {
        attr_custom_byteAlignedString version;
    }
    bit(1) has_viewBox;
    if(has_viewBox) {
        attr_viewBox viewBox;
    }
    attr_custom_valueWithUnits width;
    bit(1) has_zoomAndPan;
    if(has_zoomAndPan) {
        attr_zoomAndPan zoomAndPan;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class element_cursorManager {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_x;
    if(has_x) {
        attr_custom_coordinate x;
    }
    bit(1) has_y;
    if(has_y) {
        attr_custom_coordinate y;
    }
    bit(1) has_href;
    if(has_href) {
        attr_custom_anyURI href;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
    object_content child0;
}
class privateElementContainer {
    bit(4) ch4;
    switch(ch4){
        case 0:
            privateAnyXMLElement anyXML;
            break;
        case 1:
            privateOpaqueElement opaque;
            break;
        case 2:
            element_any reserved;
            break;
        default:
            attr_custom_extension ext;
            break;
    }
}
class element_rectClip {
    bit(1) has_id;
    if(has_id) {
        attr_custom_ID id;
    }
    bit(1) has_rare;
    if(has_rare) {
        attr_custom_rare rare;
    }
    bit(1) has_fill;
    if(has_fill) {
        attr_custom_paint fill;
    }
    bit(1) has_stroke;
    if(has_stroke) {
        attr_custom_paint stroke;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_size;
    if(has_size) {
        attr_point size;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {

```

```

    attr_any any;
  }
  object_content child0;
}
class element_selector {
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if(has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_fill;
  if(has_fill) {
    attr_custom_paint fill;
  }
  bit(1) has_stroke;
  if(has_stroke) {
    attr_custom_paint stroke;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_choice;
  if(has_choice) {
    attr_choice choice;
  }
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_simpleLayout {
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if(has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_fill;
  if(has_fill) {
    attr_custom_paint fill;
  }
  bit(1) has_stroke;
  if(has_stroke) {
    attr_custom_paint stroke;
  }
  bit(1) has_delta;
  if(has_delta) {
    attr_point delta;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class update_NewScene {
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  element_svg child;
}
class update_RefreshScene {
  vluimsbf5 time;
  bit(1) has_attr_any;
  if(has_attr_any) {
    attr_any any;
  }
  element_svg child;
}
class update_Replace {
  bit(1) has_attributeName;
  if(has_attributeName) {
    attr_AttributeName attributeName;
  }
  bit(1) has_index;
  if(has_index) {
    attr_index index;
  }
  bit(1) has_operandAttribute;
  if(has_operandAttribute) {
    attr_AttributeName operandAttribute;
  }
  bit(1) has_operandElementId;
  if(has_operandElementId) {

```

```

    attr_custom_IDREF operandElementId;
}
attr_custom_IDREF ref;
bit(1) has_value;
if(has_value) {
    attr_custom_updateValue value;
}
bit(1) has_attr_any;
if(has_attr_any) {
    attr_any any;
}
bit(1) opt_group;
if(opt_group) {
    vluimsbf5 occl;
    for(int t=0;t<occl;t++) {
        updatable_elements child0[[t]];
    }
}
}
class update_Restore {
    attr_custom_byteAlignedString groupID;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
class update_Save {
    attr_custom_ElementAttributeList elementAttributeList;
    attr_custom_byteAlignedString groupID;
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
class update_SendEvent {
    attr_custom_event event;
    bit(1) has_intvalue;
    if(has_intvalue) {
        signedInt intvalue;
    }
    bit(1) has_pointvalue;
    if(has_pointvalue) {
        attr_point pointvalue;
    }
    attr_custom_IDREF ref;
    bit(1) has_stringvalue;
    if(has_stringvalue) {
        attr_custom_byteAlignedString stringvalue;
    }
    bit(1) has_attr_any;
    if(has_attr_any) {
        attr_any any;
    }
}
}
//*****
// start specific classes
//*****

class LASERUnit {
    LASERUnitHeader h;
    initialisations c;
    vluimsbf5 occl;
    for(int t=0;t<occl+1;t++) {
        updates child0;
    }
    bit(1) opt_group;
    if(opt_group) {
        attr_custom_extension ext;
    }
}
class LASERUnitHeader {
    bit(1) resetEncodingContext;
    bit(1) opt_group;
    if(opt_group) {
        attr_custom_extension ext;
    }
}
class initialisations {
    colorInitialisation c;
    fontInitialisation f;
    privateDataIdentifierInitialisation p;
    anyXMLInitialisation a;
    extendedInitialisation e;
}

//*****
// extensions
//*****

```

```

//for elements and updates
class element_any {
    uint(extensionIDBits) reserved;
    vluimsbf5 len; //length in bits
    bit[len] toSkip;
}

//for attributes
class attr_any {
    do {
        uint(extensionIDBits) reserved;
        vluimsbf5 len; //length in bits
        bit[len] toSkip;
        bit(1) hasNextExtension;
    } while (hasNextExtension);
}

//*****
// LASerHeader
//*****

class LASerHeader {
    uint(8) profile;
    uint(8) level;
    bit(3) reserved;
    bit(2) pointsCodec;
    bit(4) pathComponents;
    bit(1) useFullRequestHost;
    bit(1) hasTimeResolution;
    if (hasTimeResolution) {
        uint(16) timeResolution;
    }
    bit(4) colorComponentBits_minus_1;
    colorComponentBits = colorComponentBits_minus_1 + 1;
    bit(4) resolution;
    bit(5) coordBits;
    bit(4) scaleBits_minus_coordBits;
    bit(1) newSceneIndicator;
    bit(3) reserved;
    // extensionIDBits defines the number of bits of extension tags
    bit(4) extensionIDBits;
    bit(1) hasExtensionDescriptor;
    if (hasExtensionDescriptor) {
        vluimsbf5 len;
        byte[len] extensionDescriptor; //extensionDescriptor can be used by a BiM decoder has specified in Annex A.
    }
    bit(1) hasExtension;
    if (hasExtension) {
        attr_custom_extension ext;
    }
}

//*****
// Initialisation codecs
//*****

int colorIndex = -1;

class colorInitialisation {
    bit(1) hasColors;
    if (hasColors) {
        // a color table in front of each AU
        vluimsbf5 nbColors;
        for (int i = 0; i < nbColors; i++) {
            colorIndex = colorIndex + 1;
            uint(colorComponentBits) red[[colorIndex]];
            uint(colorComponentBits) green[[colorIndex]];
            uint(colorComponentBits) blue[[colorIndex]];
        }
        colorIndexBits = log2sup(colorIndex);
    }
}

int fontIndex = -1;

class fontInitialisation {
    bit(1) hasFonts;
    if (hasFonts) {
        // a font table in front of each AU
        vluimsbf5 nbFonts;
        for (int i = 0; i < nbFonts; i++) {
            fontIndex = fontIndex + 1;
            attr_custom_byteAlignedString font[[fontIndex]];
        }
        fontIndexBits = log2sup(fontIndex);
    }
}

int privateDataIdentifierIndex = -1;

```

```

class privateDataIdentifierInitialisation {
    bit(1) hasPrivateDataIdentifiers;
    if (hasPrivateDataIdentifiers) {
        // a privateDataIdentifierTable in front of each AU
        vluimsbf5 nbPrivateDataIdentifiers;
        for (int i = 0; i < nbPrivateDataIdentifiers; i++) {
            privateDataIdentifierIndex = privateDataIdentifierIndex + 1;
            // the purpose of these strings is to ensure non collision between different private data
            // examples are URNs, uuids, ...
            attr_custom_byteAlignedString privateDataIdentifier[[privateDataIdentifierIndex]];
        }
        privateDataIdentifierIndexBits = log2sup(privateDataIdentifierIndex);
    }
}

int tagIndex = -1;

class anyXMLInitialisation {
    bit(1) hasTags;
    if (hasTags) {
        // a tag table in front of each AU
        vluimsbf5 nbTags;
        for (int i = 0; i < nbTags; i++) {
            if (i == 0) {
                // tag 0 for use for priv. attrs on LASer elements
                bit(1) hasAttrs;
                if (hasAttrs) {
                    // the first bin is reserved for private attributes of LASer elements
                    // and to attributes with a privateDataIdentifier different from their parent element
                    vluimsbf5 nbAttrNames[[0]];
                    for (int t = 0; t < nbAttrNames[[0]]; t++) {
                        uint(privateDataIdentifierIndexBits) privateDataIdentifierIndex[[0]][[t]];
                        attr_custom_byteAlignedString attrName[[0]][[t]];
                    }
                }
            } else {
                tagIndex = tagIndex + 1;
                uint(privateDataIdentifierIndexBits) privateDataIdentifierIndex[[i]];
                attr_custom_byteAlignedString tag[[i]];
                bit(1) hasAttrs;
                if (hasAttrs) {
                    // each private element tag has a bin for private attributes
                    vluimsbf5 nbAttrNames[[tagIndex]];
                    for (int t = 0; t < nbAttrNames[[tagIndex]]; t++) {
                        attr_custom_byteAlignedString attrName[[tagIndex]][[t]];
                    }
                }
            }
        }
        tagIndexBits = log2sup(tagIndex);
    }
}

class extendedInitialisation {
    // stringIDTable: external string ID references
    vluimsbf5 countG;
    for (int i = 0; i < countG; i++) {
        vluimsbf5 binaryIdForThisStringID[[i]];
        attr_custom_byteAlignedString stringID[[i]];
    }
    bit(1) hasExtension;
    if (hasExtension) {
        vluimsbf5 len;
        bit[len] toSkip;
    }
}

```

12.2.2 Generic Data Types

```

class object_content {
    bit(1) opt_group;
    if (opt_group) {
        privateAttributeContainer privateAttributes;
    }
    bit(1) opt_group1;
    if (opt_group1) {
        vluimsbf5 occ1;
        for(int t=0;t<occ1;t++) {
            elements child0[[t]];
        }
    }
}

class attr_AttributeName {
    bit(1) choice;
    switch(choice) {
        case 0:
            // Enumeration: a.target{0} accumulate{1} additive{2} audio-level{3} bandwidth{4} begin{5} calcMode{6}
            children{7} choice{8} clipBegin{9} clipEnd{10} color{11} color-rendering{12} cx{13} cy{14} d{15} delta{16}

```

```

display{17} display-align{18} dur{19} editable{20} enabled{21} end{22} event{23} externalResourcesRequired{24}
fill{25} fill-opacity{26} fill-rule{27} focusable{28} font-family{29} font-size{30} font-style{31} font-variant{32}
font-weight{33} fullscreen{34} gradientUnits{35} handler{36} height{37} image-rendering{38} keyPoints{39}
keySplines{40} keyTimes{41} line-increment{42} listener.target{43} mediaCharacterEncoding{44}
mediaContentEncodings{45} mediaSize{46} mediaTime{47} nav-down{48} nav-down-left{49} nav-down-right{50} nav-left{51}
nav-next{52} nav-prev{53} nav-right{54} nav-up{55} nav-up-left{56} nav-up-right{57} observer{58} offset{59}
opacity{60} overflow{61} overlay{62} path{63} pathLength{64} pointer-events{65} points{66} preserveAspectRatio{67}
r{68} repeatCount{69} repeatDur{70} requiredExtensions{71} requiredFeatures{72} requiredFormats{73} restart{74}
rotate{75} rotation{76} rx{77} ry{78} scale{79} shape-rendering{80} size{81} solid-color{82} solid-opacity{83} stop-
color{84} stop-opacity{85} stroke{86} stroke-dasharray{87} stroke-dashoffset{88} stroke-linecap{89} stroke-
linejoin{90} stroke-miterlimit{91} stroke-opacity{92} stroke-width{93} svg.height{94} svg.width{95} syncBehavior{96}
syncBehaviorDefault{97} syncReference{98} syncTolerance{99} syncToleranceDefault{100} systemLanguage{101} text-
align{102} text-anchor{103} text-decoration{104} text-display{105} text-rendering{106} textContent{107}
transform{108} transformBehavior{109} translation{110} vector-effect{111} viewBox{112} viewport-fill{113} viewport-
fill-opacity{114} visibility{115} width{116} x{117} x1{118} x2{119} xlink:actuate{120} xlink:arcrole{121}
xlink:href{122} xlink:role{123} xlink:show{124} xlink:title{125} xlink:type{126} xml:base{127} xml:lang{128} y{129}
y1{130} y2{131} zoomAndPan{132}
    bit(8) attributeName;
    break;
    case 1:
        int max+=+2;
        for (int t=0;t<max;t++) {
            vluimsbf5 item[[t]];
        }
        break;
    }
}
class attr_accumulate {
    // Enumeration: none{0} sum{1}
    bit(1) accumulate;
}
class attr_additive {
    // Enumeration: replace{0} sum{1}
    bit(1) additive;
}
class attr_calcMode {
    // Enumeration: discrete{0} linear{1} paced{2} spline{3}
    bit(2) calcMode;
}
class attr_attributeType {
    // Enumeration: CSS{0} XML{1} auto{2}
    bit(2) attributeType;
}
class attr_smil_times {
    bit(1) choice;
    switch(choice) {
        case 0:
            vluimsbf5 max;
            for(int t=0;t<max;t++) {
                custom_smil_time item[[t]];
            }
            break;
        case 1:
            // Enumeration: indefinite {0}
            break;
    }
}
class attr_time {
    bit(1) choice;
    switch(choice) {
        case 0:
            signedInt int0;
            break;
        case 1:
            // Enumeration: auto{0} indefinite{1} media{2} none{3}
            bit(2) time;
            break;
        default:
            break;
    }
}
class signedInt {
    bit(1) sign; // 1 is negative
    vluimsbf5 value;
}
class attr_animFill {
    // Enumeration: freeze{0} remove{1}
    bit(1) animFill;
}
class attr_repeatCount {
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_fixed_16_8 fixed16_8Type0;
            break;
        case 1:
            // Enumeration: indefinite{0}
            break;
        default:
            break;
    }
}

```

```

    }
}
class attr_repeatDur {
    bit(1) choice;
    switch(choice) {
        case 0:
            vluimsbf5 dur;
            break;
        case 1:
            // Enumeration: indefinite{0}
            break;
        default:
            break;
    }
}
class attr_restart {
    // Enumeration: always{0} never{1} whenNotActive{2}
    bit(2) restart;
}
class attr_floatList {
    vluimsbf5 max;
    for (int t=0;t<max;t++) {
        attr_custom_fixed_16_8 item[[t]];
    }
}
class attr_rotate {
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_fixed_16_8 fixed16_8Type0;
            break;
        case 1:
            // Enumeration: auto{0} auto-reverse{1}
            bit(1) rotate;
            break;
        default:
            break;
    }
}
class attr_rotscatra {
    // Enumeration: rotate{0} scale{1} skewX{2} skewY{3} translate{4}
    bit(3) rotscatra;
}
class attr_syncBehavior {
    // Enumeration: canSlip{0} default{1} independent{2} locked{3}
    bit(2) syncBehavior;
}
class attr_syncTolerance {
    bit(1) choice;
    switch(choice) {
        case 0:
            vluimsbf5 vluimsbf50;
            break;
        case 1:
            // Enumeration: default{0}
            break;
    }
}
class attr_preserveAspectRatio {
    bit(1) choice;
    switch(choice) {
        case 0:
            bit(1) choice;
            switch(choice) {
                case 0:
                    // Enumeration: none{0} xMaxYMax{1} xMaxYMid{2} xMaxYMin{3} xMidYMax{4} xMidYMid{5} xMidYMin{6}
                    xMinYMax{7} xMinYMid{8} xMinYMin{9}
                    bit(4) preserveAspectRatio;
                    break;
                case 1:
                    // Enumeration: _reserved{0} defer xMaxYMax{1} defer xMaxYMid{2} defer xMaxYMin{3} defer xMidYMax{4}
                    defer xMidYMid{5} defer xMidYMin{6} defer xMinYMax{7} defer xMinYMid{8} defer xMinYMin{9}
                    bit(4) preserveAspectRatio;
                    break;
            }
            break;
        case 1:
            // Enumeration: reserved{0}
            bit(5) preserveAspectRatio;
            break;
    }
}
class attr_transformBehavior {
    // Enumeration: geometric{0} pinned{1} pinned_180{2} pinned_270{3} pinned_90{4}
    bit(4) transformBehavior;
}
class attr_gradientUnits {
    // Enumeration: objectBoundingBox{0} userSpaceOnUse{1}
    bit(1) gradientUnits;
}
}

```



```

class attr_editable {
    // Enumeration: none{0} simple{1}
    bit(1) editable;
}
class objectSame_content {
    bit(1) opt_group;
    if (opt_group) {
        vluimsbf5 occ0;
        for(int t=0;t<occ0;t++) {
            elements child0[[t]];
        }
    }
}
class attr_script {
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_byteAlignedString string0;
            break;
        case 1:
            // Enumeration: application/ecmascript{0} application/jar-archive{1}
            bit(1) script;
            break;
    }
}
class attr_overflow {
    // Enumeration: visible{0}
    bit(2) overflow;
}
class attr_overlay {
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_extension overlayExType0;
            break;
        case 1:
            // Enumeration: none{0} top{1}
            bit(1) overlay;
            break;
        default:
            break;
    }
}
class attr_defaultAction {
    // Enumeration: cancel{0} perform{1}
    bit(1) defaultAction;
}
class attr_phase {
    // Enumeration: default{0}
    bit(1) phase;
}
class attr_propagate {
    // Enumeration: continue{0} stop{1}
    bit(1) propagate;
}
class attr_index {
    vluimsbf5 value;
}
class attr_playbackOrder {
    // Enumeration: all{0} forwardOnly{1}
    bit(1) playbackOrder;
}
class attr_syncBehaviorDefault {
    // Enumeration: canSlip{0} independent{1} inherit{2} locked{3}
    bit(2) syncBehaviorDefault;
}
class attr_syncToleranceDefault {
    bit(1) choice;
    switch(choice) {
        case 0:
            vluimsbf5 vluimsbf50;
            break;
        case 1:
            // Enumeration: inherit{0}
            break;
        default:
            break;
    }
}
class attr_timeLineBegin {
    // Enumeration: onLoad{0} onStart{1}
    bit(1) timeLineBegin;
}
class attr_viewBox {
    int max=+4;
    for (int t=0;t<max;t++) {
        attr_custom_fixed_16_8 item[[t]];
    }
}
class attr_zoomAndPan {

```

```

// Enumeration: disable{0} magnify{1}
bit(1) zoomAndPan;
}
class attr_point {
int max=+2;
for (int t=0;t<max;t++) {
attr_custom_coordinate item[[t]];
}
}
class attr_choice {
bit(1) choice;
switch(choice) {
case 0:
bit(8) value;
break;
case 1:
// Enumeration: all{0} none{1}
bit(1) choice;
break;
}
}
}
class attr_coordinateList {
vluimsbf5 len;
for (int t = 0; t < len; t++) {
attr_custom_coordinate coord[[t]];
}
}
}

```

Replace 12.2.3.1.1 (ID Syntax) with:

```

class attr_custom_ID {
vluimsbf5 ID;
bit(1) reserved;
if (reserved) {
vluimsbf5 len;
bit[len] reserved;
}
}
}

```

Replace 12.2.3.1.2 (ID Semantics) with:

This class allocates a number for an id.

Replace 12.2.3.2.1 (IDREF Syntax) with:

```

class attr_custom_IDREF {
vluimsbf5 href;
bit(1) reserved;
if (reserved) {
vluimsbf5 len;
bit[len] reserved;
}
}
}

```

Replace 12.2.3.3.1 (Any URI Syntax) with:

```

class attr_custom_anyURI {
bit(1) hasUri;
if (hasUri) {
attr_custom_byteAlignedString uri;
bit(1) hasData; // for a data URL, the actual data part is sent below, the header is sent in the uri field
if (hasData) {
vluimsbf5 len;
byte[len] data;
}
}
bit(1) hasID;
if (hasID) {
attr_custom_IDREF idref;
}
bit(1) hasStreamID;
if (hasStreamID) {
attr_custom_IDREF ref;
}
}
}

```

Replace 12.2.3.4.1 (Color Syntax) with:

```

class attr_custom_paint {
bit(1) hasIndex;
if (hasIndex) {
uint(colorIndexBits) color0;
} else {
bit(2) ch2;
switch(ch2) {
case 0: // enum
//Enumeration: inherit{0} currentColor{1} none{2}

```

```

        bit(2) color;
        break;
    case 1: // URI
        attr_custom_anyURI uri;
        break;
    case 2: // System Paint Server
        attr_custom_byteAlignedString serverName;
        break;
    case 3:
        attr_custom_extension colorExType0; // extensibility
        break;
    }
}
}

```

Replace 12.2.3.5.1 (Matrix Syntax) with:

```

class custom_matrix {
    bit(1) isNotMatrix;
    if (isNotMatrix) {
        bit(1) isRef; // modification for the encoding of ref(svg[,x,y])
        if (isRef) {
            bit(1) hasXY;
            if (hasXY) {
                attr_custom_fixed_16_8 valueX;
                attr_custom_fixed_16_8 valueY;
            }
        } else {
            attr_custom_extension ext;
        }
    } else {
        bit(1) xx_yy_present;
        if (xx_yy_present) {
            uint(coordBits+scaleBits_minus_coordBits) xx;
            uint(coordBits+scaleBits_minus_coordBits) yy;
        }
        bit(1) xy_yx_present;
        if (xy_yx_present) {
            uint(coordBits+scaleBits_minus_coordBits) xy;
            uint(coordBits+scaleBits_minus_coordBits) yx;
        }
        bit(1) xz_yz_present;
        if (xz_yz_present) {
            uint(coordBits+scaleBits_minus_coordBits) xz;
            uint(coordBits+scaleBits_minus_coordBits) yz;
        }
    }
}
}

```

Replace 12.2.3.7.1 (Path Syntax) with:

```

class attr_custom_path {
    attr_custom_pointSequence seq;
    vluimsbf5 nbOfTypes;
    for (int i = 0; i < nbOfTypes; i++) {
        // Enumeration: "pathSegmentTypes" C{0} H{1} L{2} M{3} Q{4} S{5} T{6} V{7} Z{8} c{9} h{10} l{11} m{12} q{13}
        s{14} t{15} v{16} z{17}
        uint(5) type[[i]];
    }
}

```

Replace 12.2.3.7.2 (Path Semantics) with:

The class `attr_custom_path` decodes a path value.

The decoding of a path value (e.g. the `d` attribute of a path element) is achieved by decoding a sequence of points and a list of types. The sequence of point is decoded as specified in 13.2.3.8. The decoded point coordinates are absolute values. When decoding the list of types, an implicit MoveTo shall be inserted before the first decoded type.

A decoder can use the decoded coordinate values directly for rendering by changing relative drawing types (a.k.a drawing instructions) to their absolute counterpart (e.g. changing 'c' to 'C'). A decoder may reconstruct an SVG path, exactly as it was in the original document, by walking in the list of drawing types and retrieving the appropriate number of coordinates according to the drawing type. For relative types (e.g. 'c', 'l'), the associated coordinate values may be restored to their relative values using the previous coordinate values.

Replace 12.2.3.8.1 (PointSequence Syntax) with:

```

class attr_custom_pointSequence {
    vluimsbf5 nbPoints;
    uint(1) flag;
    if (flag == 0) {
        if (nbPoints < 3) {
            uint(5) bits;
            for (int i = 0; i < nbPoints; i++) {
                uint(bits) x[[i]];
                uint(bits) y[[i]];
            }
        } else {
            uint(5) bits;
            uint(bits) x[0];
            uint(bits) y[0];
            uint(5) bitsx;
            uint(5) bitsy;
            for (int i = 1; i < nbPoints; i++) {
                uint(bitsx) dx;
                uint(bitsy) dy;
                x[i] = dx + x[i-1];
                y[i] = dy + y[i-1];
            }
        }
    } else {
        if (pointsCodec == 0) { // pointsCodec is a LASerHeader attribute
            uint(4) kvalue;
            uint(5) bits;
            uint(bits) x[0];
            uint(bits) y[0];
            int XMvalue, YMvalue = 0;
            int CodeNum = 0;
            int Diff = 0;
            for (int i=1; i < nbPoints; i++) {
                // to calculate X point
                do {
                    bit(1) bitX;
                    XMvalue ++;
                } while (bitX == 0);
                const bit(1) endX = 1;
                uint(XMvalue+kvalue) INFO_dx;
                CodeNum = GetCodeNum(kvalue, XMvalue, INFO_dx);
                Diff = GetDiff(CodeNum);
                x[i] = x[i-1] + Diff;
                // to calculate Y point
                do {
                    bit(1) bitY;
                    YMvalue ++;
                } while (bitY == 0);
                const bit(1) endY = 1;
                uint(YMvalue+kvalue) INFO_dy;
                CodeNum = GetCodeNum(kvalue, YMvalue, INFO_dy);
                Diff = GetDiff(CodeNum);
                y[i] = y[i-1] + Diff;
            }
        } else {
            attr_custom_extension ext;
        }
    }
}

uint GetDiff(int codeNum){
    int diff;
    if (codeNum == 0) {
        diff = 0;
        return diff;
    } else {
        if ((codeNum%2) == 0 ) {
            diff = -codeNum/2;
            return diff;
        }
        else {
            diff = (codeNum+1)/2;
            return diff;
        }
    }
}

uint GetCodeNum(int k, int Mvalue, int INFO){
    return 2^(k+Mvalue) + INFO - 2^k ;
}

```

Replace 12.2.3.9.1 (ValueWithUnits Syntax) with:

```

class attr_custom_valueWithUnits {
    uint(32) value; // float represented as fixed point with 8 bits mantissa
    uint(3) units; // 0 no unit or px, 1 'in', 2 'cm', 3 'mm', 4 'pt', 5 'pc', 6 '%'
}

```

Replace 12.2.3.10.1 (AnimatedValues Syntax) with:

```

class attr_custom_AnimatedValues {
    uint(4) type;
    vluimsbf5 nbValue;
    for (int i=0; i < nbValue; i++) {
        bit(1) escapeFlag[[i]];
        if (escapeFlag[[i]]) {
            // case for inherit and other mixed enum+number cases
            bit(2) escapeEnum[[i]];
        } else {
            switch(type) {
                case 0: // string
                    attr_custom_byteAlignedString value[[i]];
                    break;
                case 1: // float
                    attr_custom_fixed_16_8 value[[i]];
                    break;
                case 12:
                    attr_custom_anyURI value[[i]];
                    break;
                case 2: // path
                    attr_custom_path value[[i]];
                    break;
                case 3: // pointSeq
                    attr_custom_pointSequence value[[i]];
                    break;
                case 4: // fraction
                    attr_custom_0to1float value[[i]];
                    break;
                case 5: // color
                    attr_custom_paint value[[i]];
                    break;
                case 6: // enum
                case 10: // id
                    vluimsbf5 value[[i]];
                    break;
                case 11: // font
                    vluimsbf5 j;
                    value[i] = fontTable[j];
                    break;
                case 7: // ints
                    vluimsbf5 nbInts;
                    for (int k = 0; k < nbInts; k++) {
                        vluimsbf5 value[[i]][[k]];
                    }
                    break;
                case 8: // floats
                    vluimsbf5 nbFloats;
                    for (int k = 0; k < nbFloats; k++) {
                        attr_custom_fixed_16_8 value[[i]][[k]];
                    }
                    break;
                case 9: // point
                    attr_custom_coordinate valueX[[i]];
                    attr_custom_coordinate valueY[[i]];
                    break;
                default:
                    attr_custom_extension privateData;
                    break;
            }
        }
    }
}

```

Replace 12.2.3.11.1 (AnimatedValue Syntax) with:

```

class attr_custom_AnimatedValue {
    uint(4) type;
    bit(1) escapeFlag;
    if (escapeFlag) {
        // case for inherit and other mixed enum+number cases
        bit(2) escapeEnum;
    } else {
        switch(type) {
            case 0:
                attr_custom_byteAlignedString value;
                break;
            case 1:
                attr_custom_fixed_16_8 value;
                break;
            case 12:
                attr_custom_anyURI value;
                break;
            case 2:
                attr_custom_path value;
                break;
            case 3:

```

```

        attr_custom_pointSequence value;
        break;
    case 4:
        attr_custom_0to1float value;
        break;
    case 5:
        attr_custom_paint value;
        break;
    case 6: // enum
    case 10: // id
        vluimsbf5 value;
        break;
    case 11: // font
        vluimsbf5 j;
        value = fontTable[j];
        break;
    case 7: // ints
        vluimsbf5 nbInts;
        for (int k = 0; k < nbInts; k++) {
            vluimsbf5 value[[k]];
        }
        break;
    case 8: // floats
        vluimsbf5 nbFloats;
        for (int k = 0; k < nbFloats; k++) {
            attr_custom_fixed_16_8 value[[k]];
        }
        break;
    case 9: // point
        attr_custom_coordinate valueX;
        attr_custom_coordinate valueY;
        break;
    default:
        attr_custom_extension privateData;
        break;
    }
}
}
}

```

Replace 12.2.3.15.1 (UpdateValue Syntax) with:

```

const int TYPE_boolean = 0;
const int TYPE_enum = 1;
const int TYPE_color = 2;
const int TYPE_fraction = 3;
const int TYPE_float = 4;
const int TYPE_time = 5;
const int TYPE_point = 6;
const int TYPE_matrix = 7;
const int TYPE_string = 8;
const int TYPE_points = 9;
const int TYPE_path = 10;
const int TYPE_ints = 11;
const int TYPE_floats = 12;
const int TYPE_smil_times = 13;
const int TYPE_unit = 14;
const int TYPE_index = 15;
const int TYPE_URI = 16;
const int TYPE_ID = 17;
const int TYPE_event = 18;
const int TYPE_scale = 19;
const int TYPE_coordinate = 20;
const int TYPE_keyTimes = 21;

const int updateTypeOfAttribute[] = {
    TYPE_string, // a.target
    TYPE_boolean, // accumulate
    TYPE_boolean, // additive
    TYPE_fraction, // audio-level
    TYPE_float, // bandwidth
    TYPE_smil_times, // begin
    TYPE_enum, // calcMode
    -1, // children
    TYPE_index, // choice
    TYPE_time, // clipBegin
    TYPE_time, // clipEnd
    TYPE_color, // color
    TYPE_enum, // color-rendering
    TYPE_coordinate, // cx
    TYPE_coordinate, // cy
    TYPE_path, // d
    TYPE_floats, //delta
    TYPE_enum, // display
    TYPE_enum, // display-align
    TYPE_time, // dur
    TYPE_boolean, // editable
    TYPE_boolean, // enabled
    TYPE_smil_times, // end
    TYPE_event, // event
}

```

```

TYPE boolean, // externalResourcesRequired
TYPE_color, // fill
TYPE_fraction, // fill-opacity
TYPE_enum, // fill-rule
TYPE_enum, // focusable
TYPE_index, // font-family
TYPE_float, // font-size
TYPE_enum, // font-style
TYPE_enum, // font-variant
TYPE_enum, // font-weight
TYPE_boolean, // fullscreen
TYPE_enum, // gradient-units
TYPE_ID, // handler
TYPE_coordinate, // height
TYPE_enum, // image-rendering
TYPE_floats, // keyPoints
TYPE_floats, // keySplines
TYPE_keyTimes, // keyTimes
TYPE_float, // line-increment
TYPE_ID, // listener.target
TYPE_string, // mediaCharacterEncoding
TYPE_string, // mediaContentEncodings
TYPE_float, // mediaSize
TYPE_time, // mediaTime
TYPE_ID, // nav-down
TYPE_ID, // nav-down-left
TYPE_ID, // nav-down-right
TYPE_ID, // nav-left
TYPE_ID, // nav-next
TYPE_ID, // nav-prev
TYPE_ID, // nav-right
TYPE_ID, // nav-up
TYPE_ID, // nav-up-left
TYPE_ID, // nav-up-right
TYPE_ID, // observer
TYPE_fraction, // offset
TYPE_fraction, // opacity
TYPE_enum, // overflow
TYPE_enum, // overlay
TYPE_path, // path
TYPE_float, // pathLength
TYPE_enum, // pointer-events
TYPE_points, // points
-1, // preserveAspectRatio
TYPE_coordinate, // r
TYPE_float, // repeatCount
TYPE_time, // repeatDur
TYPE_string, // requiredExtensions
TYPE_ints, // requiredFeatures
TYPE_string, // requiredFormats
TYPE_enum, // restart
TYPE_floats, // rotate
TYPE_float, // rotation
TYPE_coordinate, // rx
TYPE_coordinate, // ry
TYPE_scale, // scale
TYPE_enum, // shape-rendering
TYPE_point, // size
TYPE_color, // solid-color
TYPE_fraction, // solid-opacity
TYPE_color, // stop-color
TYPE_fraction, // stop-opacity
TYPE_color, // stroke
TYPE_floats, // stroke-dasharray
TYPE_float, // stroke-dashoffset
TYPE_enum, // stroke-linecap
TYPE_enum, // stroke-linejoin
TYPE_float, // stroke-miterlimit
TYPE_fraction, // stroke-opacity
TYPE_float, // stroke-width
TYPE_unit, // svg.height
TYPE_unit, // svg.width
TYPE_enum, // syncBehavior
TYPE_enum, // syncBehaviorDefault
TYPE_URI, // syncReference
TYPE_float, // syncTolerance
TYPE_float, // syncToleranceDefault
TYPE_string, // systemLanguage
TYPE_enum, // text-align
TYPE_enum, // text-anchor
TYPE_enum, // text-decoration
TYPE_enum, // text-rendering
TYPE_string, // textContent
TYPE_matrix, // transform
TYPE_enum, // transformBehavior
TYPE_point, // translation
TYPE_enum, // vector-effect,
TYPE_floats, // viewBox
TYPE_color, // viewport-fill
TYPE_fraction, // viewport-fill-opacity

```

```

TYPE_enum, // visibility
TYPE_coordinate, // width
TYPE_coordinate, // x
TYPE_coordinate, // x1
TYPE_coordinate, // x2
TYPE_enum, // xlink:actuate
TYPE_string, // xlink:arcrole
TYPE_URI, // xlink:href
TYPE_string, // xlink:role
TYPE_enum, // xlink:show
TYPE_string, // xlink:title
TYPE_enum, // xlink:type
TYPE_string, // xml:base
TYPE_string, // xml:lang
TYPE_coordinate, // y
TYPE_coordinate, // y1
TYPE_coordinate, // y2
TYPE_boolean // zoomAndPan
};

const int ITYPE_point = 0;
const int ITYPE_int = 1;
const int ITYPE_float = 2;
const int ITYPE_keyTime = 3;
const int ITYPE_0tol = 4;
const int ITYPE_smil_time = 5;

const int indexedTypeOfAttribute[] = {
-1, // a.target
-1, // accumulate
-1, // additive
-1, // audio-level
-1, // bandwidth
ITYPE_smil_time, // begin
-1, // calcMode
-1, // children
-1, // choice
-1, // clipBegin
-1, // clipEnd
-1, // color
-1, // color-rendering
-1, // cx
-1, // cy
-1, // d
-1, // delta
-1, // display
-1, // display-align
-1, // dur
-1, // editable
-1, // enabled
ITYPE_smil_time, // end
-1, // event
-1, // externalResourcesRequired
-1, // fill
-1, // fill-opacity
-1, // fill-rule
-1, // focusable
-1, // font-family
-1, // font-size
-1, // font-style
-1, // font-variant
-1, // font-weight
-1, // fullscreen
-1, // gradient-units
-1, // handler
-1, // height
-1, // image-rendering
ITYPE_0tol, // keyPoints
ITYPE_float, // keySplines
ITYPE_keyTime, // keyTimes
-1, // line-increment
-1, // listener.target
-1, // mediaCharacterEncoding
-1, // mediaContentEncodings
-1, // mediaSize
-1, // mediaTime
-1, // nav-down
-1, // nav-down-left
-1, // nav-down-right
-1, // nav-left
-1, // nav-next
-1, // nav-prev
-1, // nav-right
-1, // nav-up
-1, // nav-up-left
-1, // nav-up-right
-1, // observer
-1, // offset
-1, // opacity
-1, // overflow

```



```

-1, // overlay
-1, // path
-1, // pathLength
-1, // pointer-events
ITYPE_point, // points
-1, // preserveAspectRatio
-1, // r
-1, // repeatCount
-1, // repeatDur
-1, // requiredExtensions
ITYPE_int, // requiredFeatures
-1, // requiredFormats
-1, // restart
-1, // rotate
-1, // rotation
-1, // rx
-1, // ry
-1, // scale
-1, // shape-rendering
-1, // size
-1, // solid-color
-1, // solid-opacity
-1, // stop-color
-1, // stop-opacity
-1, // stroke
ITYPE_float, // stroke-dasharray
-1, // stroke-dashoffset
-1, // stroke-linecap
-1, // stroke-linejoin
-1, // stroke-miterlimit
-1, // stroke-opacity
-1, // stroke-width
-1, // svg.height
-1, // svg.width
-1, // syncBehavior
-1, // syncBehaviorDefault
-1, // syncReference
-1, // syncTolerance
-1, // syncToleranceDefault
-1, // systemLanguage
-1, // text-align
-1, // text-anchor
-1, // text-decoration
-1, // text-rendering
-1, // textContent
-1, // transform
-1, // transformBehavior
-1, // translation
-1, // vector-effect
ITYPE_float, // viewBox
-1, // viewport-fill
-1, // viewport-fill-opacity
-1, // visibility
-1, // width
-1, // x
-1, // x1
-1, // x2
-1, // xlink:actuate
-1, // xlink:arcrole
-1, // xlink:href
-1, // xlink:role
-1, // xlink:show
-1, // xlink:title
-1, // xlink:type
-1, // xml:base
-1, // xml:lang
-1, // y
-1, // y1
-1, // y2
-1 // zoomAndPan
};

```

```

class attr_custom_updateValue {
    if (updateHasIndex) { // is the hasIndex bit set in the update ? false if not in an update
        // indexed
        switch(indexedTypeOfAttribute[attributeIndex]) {
            case ITYPE_point:
                attr_custom_coordinate pointValueX;
                attr_custom_coordinate pointValueY;
                break;
            case ITYPE_int:
                byte intValue;
                break;
            case ITYPE_float:
                attr_custom_fixed_16_8 floatValue;
                break;
            case ITYPE_keyTime:
                bit(1) hasShort;
                if (hasShort) {

```

```

        bit(1) isZero;
        if (isZero) {
            timevalue = 0;
        } else {
            timevalue = 1;
        }
    } else {
        uint(12) timevalue; // 0 to 1 float encoded on 12 bits
    }
    break;
case ITYPE_smil_time:
    custom_smil_time timevalue;
    break;
case ITYPE_0to1:
    attr_custom_0to1float valueX;
    attr_custom_0to1float valueY;
    break;
default:
    attr_custom_extension privateData;
    break;
}
} else {
switch(updateTypeOfAttribute[attributeIndex]) {
case TYPE_boolean:
    bit(1) booleanvalue;
    break;
case TYPE_enum:
    bit(1) isDefaultValue; // default value (inherit, auto, default ...)
    if (!isDefaultValue) {
        vluimsbf5 enumValue;
    }
    break;
case TYPE_index:
    bit(1) isDefaultValue; // default value (inherit, auto, default ...)
    if (!isDefaultValue) {
        bit(1) escapeFlag;
        if (escapeFlag) {
            // case for mixed enum+number cases
            bit(2) escapeEnumVal;
        } else {
            vluimsbf5 indexValue;
        }
    }
    break;
case TYPE_ID:
    bit(1) isDefaultValue; // default value (inherit, auto, default ...)
    if (!isDefaultValue) {
        bit(1) escapeFlag;
        if (escapeFlag) {
            // case for mixed enum+ID cases
            bit(2) escapeEnumVal;
        } else {
            vluimsbf5 IDValue;
        }
    }
    break;
case TYPE_color:
    attr_custom_paint colorValue;
    break;
case TYPE_fraction:
    bit(1) isDefaultValue; // default value (inherit, auto, default ...)
    if (!isDefaultValue) {
        attr_custom_0to1float fractionValue;
    }
    break;
case TYPE_float:
    bit(1) isDefaultValue; // default value (inherit, auto, default ...)
    if (!isDefaultValue) {
        bit(1) escapeFlag;
        if (escapeFlag) {
            // case for mixed enum+float cases
            bit(2) escapeEnumVal;
        } else {
            attr_custom_fixed_16_8 floatValue;
        }
    }
    break;
case TYPE_time:
    attr_time timeValue;
    break;
case TYPE_point:
    attr_custom_coordinate pointValueX;
    attr_custom_coordinate pointValueY;
    break;
case TYPE_matrix:
    custom_matrix matrixValue;
    break;
case TYPE_string:
    attr_custom_byteAlignedString stringValue;
    break;

```

```

case TYPE_points:
    attr_custom_pointSequence pointValues;
    break;
case TYPE_path:
    attr_custom_path pathValue;
    break;
case TYPE_ints:
    vluimsbf5 nb;
    for (int i = 0; i < nb; i++) {
        uint(8) intValue[[i]];
    }
    break;
case TYPE_floats:
    bit(1) isDefaultValue; // only for stroke-dasharray inherit
    if (!isDefaultValue) {
        bit(1) escapeFlag;
        if (escapeFlag) { // only for stroke-dasharray none
            bit(2) escapeEnumVal;
        } else {
            vluimsbf5 nb1;
            for (int i = 0; i < nb1; i++) {
                attr_custom_fixed_16_8 floatValue[[i]];
            }
        }
    }
    break;
case TYPE_keyTimes:
    attr_custom_fraction12List timeValues;
    break;
case TYPE_smil_times:
    attr_smil_times timeValues1;
    break;
case TYPE_unit:
    attr_custom_valueWithUnits unitValue;
    break;
case TYPE_URI:
    attr_custom_anyURI uri;
    break;
case TYPE_event:
    attr_custom_event event;
    break;
case TYPE_scale:
    attr_custom_fixed_16_8 scaleX;
    attr_custom_fixed_16_8 scaleY;
    break;
case TYPE_coordinate:
    attr_custom_coordinate value;
    break;
default:
    attr_custom_extension privateData;
    break;
}
}
}

```

Replace 12.2.3.16.1 (Fraction12List Syntax) with:

```

class attr_custom_fraction12List {
    vluimsbf5 len;
    for (int i = 0; i < len; i++) {
        bit(1) hasShort;
        if (hasShort) {
            bit(1) isZero;
            if (isZero) {
                float[i] = 0;
            } else {
                float[i] = 1;
            }
        } else {
            uint(12) float[[i]]; // 0 to 1 float encoded on 12 bits
        }
    }
}

```

Replace 12.2.3.17.1 (Event Syntax) with:

```

class attr_custom_event {
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_byteAlignedString string0;
            break;
        case 1:
            // Enumeration: abort{0} accessKey{1} activate{2} activatedEvent{3} beginEvent{4} click{5} deactivatedEvent{6}
            endEvent{7} error{8} executionTime{9} focusin{10} focusout{11} keydown{12} keyup{13} load{14} longAccessKey{15}
            mousedown{16} mousemove{17} mouseout{18} mouseover{19} mouseup{20} pause{21} pausedEvent{22} play{23}
            repeatEvent{24} resize{25} resumedEvent{26} scroll{27} textinput{28} unload{29} zoom{30}
    }
}

```

```

    bit(6) event;
    switch (event) {
    case 1: //accessKey
    case 12: //longAccessKey
        // Enumeration: *{0} 0{1} 1{2} 2{3} 3{4} 4{5} 5{6} 6{7} 7{8} 8{9} 9{10} ANY_KEY{11} DOWN{12} FIRE{13}
        LEFT{14} NO_KEY{15} RIGHT{16} SHARP{17} SOFT_KEY_1{18} SOFT_KEY_2{19} UP{20}
        vlui5sb5f5 keyCode;
        break;
    }
    break;
}
}

```

Replace 12.2.3.18.1 (RareAttributes Syntax) with:

```

//indexes for rare attributes
const int RARE_CLASS = 0;
const int RARE_AUDIO_LEVEL = 1;
const int RARE_COLOR = 2;
const int RARE_COLOR_RENDERING = 3;
const int RARE_DISPLAY = 4;
const int RARE_DISPLAY_ALIGN = 5;
const int RARE_FILL_OPACITY = 6;
const int RARE_FILL_RULE = 7;
const int RARE_IMAGE_RENDERING = 8;
const int RARE_LINE_INCREMENT = 9;
const int RARE_POINTER_EVENTS = 10;
const int RARE_SHAPE_RENDERING = 11;
const int RARE_SOLID_COLOR = 12;
const int RARE_SOLID_OPACITY = 13;
const int RARE_STOP_COLOR = 14;
const int RARE_STOP_OPACITY = 15;
const int RARE_STROKE_DASHARRAY = 16;
const int RARE_STROKE_DASHOFFSET = 17;
const int RARE_STROKE_LINECAP = 18;
const int RARE_STROKE_LINEJOIN = 19;
const int RARE_STROKE_MITERLIMIT = 20;
const int RARE_STROKE_OPACITY = 21;
const int RARE_STROKE_WIDTH = 22;
const int RARE_TEXT_ANCHOR = 23;
const int RARE_TEXT_RENDERING = 24;
const int RARE_VIEWPORT_FILL = 25;
const int RARE_VIEWPORT_FILL_OPACITY = 26;
const int RARE_VECTOR_EFFECT = 27;
const int RARE_VISIBILITY = 28;
const int RARE_REQUIREDEXTENSIONS = 29;
const int RARE_REQUIREDFEATURES = 30;
const int RARE_REQUIREDFORMATS = 31;
const int RARE_SYSTEMLANGUAGE = 32;
const int RARE_XML_BASE = 33;
const int RARE_XML_LANG = 34;
const int RARE_XML_SPACE = 35;
const int RARE_FOCUSNEXT = 36;
const int RARE_FOCUSNORTH = 37;
const int RARE_FOCUSNORTHEAST = 38;
const int RARE_FOCUSNORTHWEST = 39;
const int RARE_FOCUSPREV = 40;
const int RARE_FOCUSSOUTH = 41;
const int RARE_FOCUSSOUTHEAST = 42;
const int RARE_FOCUSSOUTHWEST = 43;
const int RARE_FOCUSWEST = 44;
const int RARE_FOCUSABLE = 45;
const int RARE_FOCUSEAST = 46;
const int RARE_TRANSFORM = 47;
const int RARE_TEXT_DECORATION = 48; //DCOR
const int RARE_EXTENSION = 49; //DCOR
const int RARE_FONT_VARIANT = 50;
const int RARE_FONT_FAMILY = 51;
const int RARE_FONT_SIZE = 52;
const int RARE_FONT_STYLE = 53;
const int RARE_FONT_WEIGHT = 54;
const int RARE_HREF_TITLE = 55;
const int RARE_HREF_TYPE = 56;
const int RARE_HREF_ROLE = 57;
const int RARE_HREF_ARCROLE = 58;
const int RARE_HREF_ACTUATE = 59;
const int RARE_HREF_SHOW = 60;
const int RARE_END = 61;
const int RARE_MAX = 62;
const int RARE_MIN = 63;

class attr_custom_rare {
    uint(6) nbOfAttributes;
    for(int i = 0; i < nbOfAttributes; i++) {
        uint(6) attributeRARE;
        switch(attributeRARE) {
            case RARE_CLASS:
                attr_custom_byteAlignedString class;
                break;

```

```

case RARE_AUDIO_LEVEL:
    attr_custom_0tofloat audio-level;
    break;
case RARE_COLOR:
    attr_custom_paint color;
    break;
case RARE_COLOR_RENDERING:
    // Enumeration: auto{0} inherit{1} optimizeQuality{2} optimizeSpeed{3}
    bit(2) color-rendering;
    break;
case RARE_DISPLAY:
    // Enumeration: block{0} compact{1} inherit{2} inline{3} inline-table{4} list-item{5} marker{6} none{7} run-
in{8} table{9} table-caption{10} table-cell{11} table-column{12} table-column-group{13} table-footer-group{14}
table-header-group{15} table-row{16} table-row-group{17}
    bit(5) display;
    break;
case RARE_DISPLAY_ALIGN:
    // Enumeration: after{0} auto{1} before{2} center{3} inherit{4}
    bit(3) displayAlign;
    break;
case RARE_FILL_OPACITY:
    attr_custom_0tofloat f;
    break;
case RARE_FILL_RULE:
    // Enumeration: evenodd{0} inherit{1} nonzero{2}
    bit(2) fill-rule;
    break;
case RARE_IMAGE_RENDERING:
    // Enumeration: auto{0} inherit{1} optimizeQuality{2} optimizeSpeed{3}
    bit(2) image-rendering;
    break;
case RARE_LINE_INCREMENT:
    bit(1) choice;
    switch(choice) {
        case 0:
            attr_custom_fixed_16_8 line-increment;
            break;
        case 1:
            // Enumeration: auto{0} inherit{1}
            bit(1) line-increment;
            break;
        default:
            break;
    }
    break;
case RARE_POINTER_EVENTS:
    // Enumeration: all{0} boundingBox{1} fill{2} inherit{3} none{4} painted{5} stroke{6} visible{7}
visibleFill{8} visiblePainted{9} visibleStroke{10}
    bit(4) pointer-events;
    break;
case RARE_SHAPE_RENDERING:
    // Enumeration: auto{0} crispEdges{1} geometricPrecision{2} inherit{3} optimizeSpeed{4}
    bit(3) shape-rendering;
    break;
case RARE_SOLID_COLOR:
    attr_custom_paint color;
    break;
case RARE_SOLID_OPACITY:
    attr_custom_0tofloat opacity;
    break;
case RARE_STOP_COLOR:
    attr_custom_paint color;
    break;
case RARE_STOP_OPACITY:
    attr_custom_0tofloat opacity;
    break;
case RARE_STROKE_DASHARRAY:
    bit(1) isInherit;
    if (!isInherit) {
        vluimsbf5 max;
        for (int i = 0; i < max; i++) {
            attr_custom_fixed_16_8 dash[[i]];
        }
    }
    break;
case RARE_STROKE_DASHOFFSET:
    custom_fixed_16_8i dashOffset;
    break;
case RARE_STROKE_LINECAP:
    // Enumeration: butt{0} inherit{1} round{2} square{3}
    bit(2) stroke-linecap;
    break;
case RARE_STROKE_LINEJOIN:
    // Enumeration: bevel{0} inherit{1} miter{2} round{3}
    bit(2) stroke-linejoin;
    break;
case RARE_STROKE_MITERLIMIT:
    custom_fixed_16_8i miterlimit;
    break;
case RARE_STROKE_OPACITY:

```

```

    attr_custom_0tofloat opacity;
    break;
case RARE_STROKE_WIDTH:
    custom_fixed_16_8i width;
    break;
case RARE_TEXT_ANCHOR:
    // Enumeration: end{0} inherit{1} middle{2} start{3}
    bit(2) textAnchor;
case RARE_TEXT_RENDERING:
    // Enumeration: auto{0} geometricPrecision{1} inherit{2} optimizeLegibility{3} optimizeSpeed{4}
    bit(3) text-rendering;
    break;
case RARE_VIEWPORT_FILL:
    attr_custom_paint color;
    break;
case RARE_VIEWPORT_FILL_OPACITY:
    attr_custom_0tofloat opacity;
    break;
case RARE_VECTOR_EFFECT:
    // Enumeration: default{0} inherit{1} non-scaling-stroke{2}
    bit(4) vector-effect;
    break;
case RARE_VISIBILITY:
    // Enumeration: hidden{0} inherit{1} visible{2}
    bit(2) visibility;
    break;
case RARE_REQUIREDEXTENSIONS:
    attr_custom_byteAlignedString string;
    break;
case RARE_REQUIREDFEATURES:
    vluimsbf5 max;
    for (int t=0;t<max;t++) {
        // Enumeration:
        http://www.w3.org/Graphics/SVG/feature/1.2/#Animation{0}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Audio{1}
        http://www.w3.org/Graphics/SVG/feature/1.2/#ComposedVideo{2}
        http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessing{3}
        http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessingAttribute{4}
        http://www.w3.org/Graphics/SVG/feature/1.2/#CoreAttribute{5}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Discard{6}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Extensibility{7}
        http://www.w3.org/Graphics/SVG/feature/1.2/#ExternalResourcesRequired{8}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Font{9}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Gradient{10}
        http://www.w3.org/Graphics/SVG/feature/1.2/#GraphicsAttribute{11}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Handler{12}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Hyperlinking{13}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Image{14}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Listener{15}
        http://www.w3.org/Graphics/SVG/feature/1.2/#MediaAttribute{16}
        http://www.w3.org/Graphics/SVG/feature/1.2/#NavigationAttribute{17}
        http://www.w3.org/Graphics/SVG/feature/1.2/#OpacityAttribute{18}
        http://www.w3.org/Graphics/SVG/feature/1.2/#PaintAttribute{19}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Prefetch{20}
        http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-all{21}
        http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-animated{22}
        http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static{23}
        http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static-DOM{24}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Scripting{25}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Shape{26}
        http://www.w3.org/Graphics/SVG/feature/1.2/#SolidColor{27}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Structure{28}
        http://www.w3.org/Graphics/SVG/feature/1.2/#Text{29}
        http://www.w3.org/Graphics/SVG/feature/1.2/#TextFlow{30}
        http://www.w3.org/Graphics/SVG/feature/1.2/#TimedAnimation{31}
        http://www.w3.org/TR/SVG11/feature#Animation{32}
        http://www.w3.org/TR/SVG11/feature#AnimationEventsAttribute{33}
        http://www.w3.org/TR/SVG11/feature#BasicClip{34}
        http://www.w3.org/TR/SVG11/feature#BasicFilter{35}
        http://www.w3.org/TR/SVG11/feature#BasicFont{36}
        http://www.w3.org/TR/SVG11/feature#BasicGraphicsAttribute{37}
        http://www.w3.org/TR/SVG11/feature#BasicPaintAttribute{38}
        http://www.w3.org/TR/SVG11/feature#BasicStructure{39}
        http://www.w3.org/TR/SVG11/feature#BasicText{40}
        http://www.w3.org/TR/SVG11/feature#Clip{41}
        http://www.w3.org/TR/SVG11/feature#ColorProfile{42}
        http://www.w3.org/TR/SVG11/feature#ConditionalProcessing{43}
        http://www.w3.org/TR/SVG11/feature#ContainerAttribute{44}
        http://www.w3.org/TR/SVG11/feature#CoreAttribute{45}
        http://www.w3.org/TR/SVG11/feature#Cursor{46}
        http://www.w3.org/TR/SVG11/feature#DocumentEventsAttribute{47}
        http://www.w3.org/TR/SVG11/feature#Extensibility{48}
        http://www.w3.org/TR/SVG11/feature#ExternalResourcesRequired{49}
        http://www.w3.org/TR/SVG11/feature#Filter{50}
        http://www.w3.org/TR/SVG11/feature#Font{51}
        http://www.w3.org/TR/SVG11/feature#Gradient{52}
        http://www.w3.org/TR/SVG11/feature#GraphicalEventsAttribute{53}

```

```

http://www.w3.org/TR/SVG11/feature#GraphicsAttribute{57}
http://www.w3.org/TR/SVG11/feature#Hyperlinking{58}
http://www.w3.org/TR/SVG11/feature#Image{59}
http://www.w3.org/TR/SVG11/feature#Marker{60}
http://www.w3.org/TR/SVG11/feature#Mask{61}
http://www.w3.org/TR/SVG11/feature#OpacityAttribute{62}
http://www.w3.org/TR/SVG11/feature#PaintAttribute{63}
http://www.w3.org/TR/SVG11/feature#Pattern{64}
http://www.w3.org/TR/SVG11/feature#SVG{65}
http://www.w3.org/TR/SVG11/feature#SVG-animation{66}
http://www.w3.org/TR/SVG11/feature#SVG-dynamic{67}
http://www.w3.org/TR/SVG11/feature#SVG-static{68}
http://www.w3.org/TR/SVG11/feature#SVGDOM{69}
http://www.w3.org/TR/SVG11/feature#SVGDOM-animation{70}
http://www.w3.org/TR/SVG11/feature#SVGDOM-dynamic{71}
http://www.w3.org/TR/SVG11/feature#SVGDOM-static{72}
http://www.w3.org/TR/SVG11/feature#Script{73}
http://www.w3.org/TR/SVG11/feature#Shape{74}
http://www.w3.org/TR/SVG11/feature#Structure{75}
http://www.w3.org/TR/SVG11/feature#Style{76}
http://www.w3.org/TR/SVG11/feature#Text{77}
http://www.w3.org/TR/SVG11/feature#View{78}
http://www.w3.org/TR/SVG11/feature#ViewportAttribute{79}
http://www.w3.org/TR/SVG11/feature#XlinkAttribute{80}
http://www.w3.org/TR/SVGMobile/Basic/feature#SVGBasicDomCore{81}
http://www.w3.org/TR/SVGMobile/Basic/feature#SVGBasicDomExtended{82}
http://www.w3.org/TR/SVGMobile/Basic/feature#all{83}
http://www.w3.org/TR/SVGMobile/Basic/feature#base{84}
http://www.w3.org/TR/SVGMobile/Basic/feature#css{85}
http://www.w3.org/TR/SVGMobile/Basic/feature#interactivity{86}
http://www.w3.org/TR/SVGMobile/Tiny/feature#all{87}
http://www.w3.org/TR/SVGMobile/Tiny/feature#base{88}
http://www.w3.org/TR/SVGMobile/Tiny/feature#interactivity{89}
org.w3c.dom.svg{90} org.w3c.dom.svg.all{91}
org.w3c.dom.svg.animation{92}
org.w3c.dom.svg.dynamic{93}
org.w3c.dom.svg.static{94}
org.w3c.svg{95}
org.w3c.svg.all{96}
org.w3c.svg.animation{97}
org.w3c.svg.dynamic{98}
org.w3c.svg.static{99}
    bit(8) feature[{}];
}
break;
case RARE_REQUIREDFORMATS:
    attr_custom_byteAlignedString string;
    break;
case RARE_SYSTEMLANGUAGE:
    attr_custom_byteAlignedString string;
    break;
case RARE_XML_BASE:
    attr_custom_byteAlignedString string;
    break;
case RARE_XML_LANG:
    attr_custom_byteAlignedString string;
    break;
case RARE_XML_SPACE:
    // Enumeration: default{0} preserve{1}
    bit(1) space;
    break;
case RARE_FOCUSNEXT:
    custom_focus focusNext;
    break;
case RARE_FOCUSNORTH:
    custom_focus focusNorth;
    break;
case RARE_FOCUSNORTHEAST:
    custom_focus focusEast;
    break;
case RARE_FOCUSNORTHWEST:
    custom_focus focusWest;
    break;
case RARE_FOCUSPREV:
    custom_focus focusPrev;
    break;
case RARE_FOCUSSOUTH:
    custom_focus focusSouth;
    break;
case RARE_FOCUSSOUTHEAST:
    custom_focus focusSouthEast;
    break;
case RARE_FOCUSSOUTHWEST:
    custom_focus focusSouthWes;
    break;
case RARE_FOCUSWEST:
    custom_focus focusWest;
    break;
case RARE_FOCUSABLE:
    // Enumeration: auto{0} false{1} true{2}

```

```

        bit(2) focusable;
        break;
    case RARE_FOCUSEAST:
        custom_focus focusEast;
        break;
    case RARE_TRANSFORM:
        custom_matrix matrix;
        break;
    case RARE_FONT_VARIANT:
        // Enumeration: inherit{0} normal{1} small-caps{2}
        bit(2) font-variant;
        break;
    case RARE_FONT_FAMILY:
        bit(1) isInherit;
        if (!isInherit) {
            uint(fontIndexBits) fontIndex;
        }
        break;
    case RARE_FONT_SIZE:
        custom_fixed_16_8i size;
        break;
    case RARE_FONT_STYLE:
        // Enumeration: inherit{0} italic{1} normal{2} oblique{3}
        bit(3) font-style;
        break;
    case RARE_TEXT_DECORATION:
        // Enumeration: LAsER_EmBOSS{0} LAsER_ENGRAVE{1} LAsER_LEFTDROPSHADOW{2} LAsER_OUTLINE{3}
        LAsER_RIGHTDROPSHADOW{4} inherit{5} none{6} underline{7}
        bit(4) text-decoration;
        break;
    case RARE_FONT_WEIGHT:
        // Enumeration: 100{0} 200{1} 300{2} 400{3} 500{4} 600{5} 700{6} 800{7} 900{8} bold{9} bolder{10}
        inherit{11} lighter{12} normal{13}
        bit(4) font-weight;
        break;
    case RARE_HREF_TITLE:
        attr_custom_byteAlignedString title;
        break;
    case RARE_HREF_TYPE:
        // Enumeration: simple{0}
        bit(3) type;
        break;
    case RARE_HREF_ROLE:
        attr_custom_anyURI role;
        break;
    case RARE_HREF_ARCROLE:
        attr_custom_anyURI arcrole;
        break;
    case RARE_HREF_ACTUATE:
        // Enumeration: onLoad{0} onRequest{1}
        bit(2) actuate;
        break;
    case RARE_HREF_SHOW:
        // Enumeration: embed{0} new{1} other{2} replace{3}
        bit(3) show;
        break;
    case RARE_END:
        attr_smil_times time;
        break;
    case RARE_MAX:
        attr_time time;
        break;
    case RARE_MIN:
        attr_time time;
        break;
    case RARE_EXTENSION:
        attr_any ext;
        break;
    }
}
}
class custom_focus {
    bit(1) isEnum;
    if (isEnum) {
        //Enumeration: auto{0} self{1};
        bit(1) enum;
    } else {
        attr_custom_IDREF id;
    }
}
class custom_fixed_16_8i {
    bit(1) isInherit;
    if (!isInherit) {
        int(24) float;
    }
}
}

```


Replace 12.2.3.19.1 (Time Syntax) with:

```
class custom_smil_time {
    bit(1) hasEvent;
    if (hasEvent) {
        bit(1) hasIdentifier;
        if (hasIdentifier) {
            attr_custom_IDREF idref;
        }
        attr_custom_event event;
    }
    bit(1) hasClock;
    if (hasClock) {
        bit(1) sign; //1 is negative
        vluimsbf5 clock;
    }
}
```

Replace 12.2.3.19.2 (Time Semantics) with:

The class `custom_smil_time` decodes a single time value in the begin and end attribute.

Replace 12.2.3.21 (IDREFS) with:

12.2.3.21 Element-Attribute pair

12.2.3.21.1 Syntax

```
class attr_custom_ElementAttributeList {
    vluimsbf5 nbIds;
    for (int i = 0; i < nbIds; i++) {
        attr_custom_IDREF ref[[i]];
        attr_AttributeName attr[[i]];
    }
}
```

12.2.3.21.2 Semantics

The class `attr_custom_ElementAttributeList` holds a list of pairs of element ID and attribute ID.

Replace 12.2.3.22 (UpdateList) with:

12.2.3.22 ByteAlignment

12.2.3.22.1 Syntax

```
class attr_custom_align {
    aligned bit(0) dummy;
}
```

12.2.3.22.2 Semantics

This class executes byte-alignment.

Replace 12.2.3.23.1 (Extension Syntax) with:

```
class attr_custom_extension {
    vluimsbf5 len;
    byte[len] privateData;
}
```

In subclause 13.3.1 "Classification Scheme" replace:

```
<Term termID="6">
    <Name xml:lang="en">UpdateList</Name>
    <Definition xml:lang="en">Encodes a list of updates with byte alignment</Definition>
</Term>
```

with:

```
<Term termID="6">
    <Name xml:lang="en">ByteAlign</Name>
    <Definition xml:lang="en">Encodes byte alignment</Definition>
</Term>
```

In subclause 13.4.5 replace:

Syntax is defined by the class `attr_custom_Element_IDREFS` given in subclause 12.2.3.21
with:

Syntax is defined by the class `attr_custom_ElementAttributeList` given in subclause 12.2.3.21.

Replace subclause 13.4.6 with:

13.4.6 ByteAlign Codec

This codec is mapped to the type `urn:mpeg:mpeg4:LASeR:types:2005:alignType`.
Syntax is defined by the class `attr_custom_align` given in subclause 12.2.3.22.1.

In subclause 13.4.14, replace:

Syntax is defined by the class `attr_custom_fixed16_8` given in subclause 12.2.3.13.
with:

Syntax is defined by the class `attr_custom_fixed_16_8` given in subclause 12.2.3.13.

Replace content of subclause 13.4.22 with:

This codec is mapped to the type `urn:mpeg:mpeg4:LASeR:types:2005:smilTimeType` type of the LASeR encoding schema. Syntax is defined by the class `custom_smil_time` in subclause 12.2.3.19.

In subclause 13.6 "Decoding Process", insert the following paragraph at the end of the subclause:

Decoding of Encoding-related Elements

Each element prefixed by 'encoding-' shall be transformed by removing it from the document tree together with its content.

In subclause 13.6, replace:

Considerations for sameX constructs and script usage: In the following sequence of elements "X, script, and X", the second X element cannot be encoded using the sameX construct if the script element uses LASeR Updates, while it can if the script uses ECMAScript.

with:

Considerations for sameX constructs and conditional usage: In the following sequence of elements "X, conditional, and X", the second X element cannot be encoded using the sameX construct.

Replace electronic attachment LASeRML\saf1.xsd and intermediateXML\saf1.xsd with the attached versions.

Replace electronic attachment LASeRML\laser-datatypes1.xsd and intermediateXML\laser-datatypes1.xsd with the attached version.

Replace electronic attachment LASeRML\laser1.xsd and intermediateXML\laser1.xsd with the attached version.

Replace electronic attachment LASeRML\svg1.xsd and intermediateXML\svg1.xsd with the attached version.

Replace electronic attachment LASeRML\ev.xsd and intermediateXML\ev.xsd with the attached version.

Replace Electronic attachment SDL.txt with attached SDL_cor1.txt.