INTERNATIONAL STANDARD

ISO/IEC 14496-20

First edition
2006-06-15

AMENDMENT 1
2008-03-01

Corrected version
2008-04-15

Information technology — Coding of audio-visual objects —

Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)

AMENDMENT 1: LASeR extensions

Technologies de l'information — Codage des objets audiovisuels —

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

AMENDEMENT 1: Extensions LASeR

Reference number
ISO/IEC 14496-20:2006/Amd.1:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 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*.

This corrected version of ISO/IEC 14496-20:2006/Amd.1:2008 incorporates the following corrections: a change to the title of the amendment and the provision of updated electronic attachments relevant to LASeR.

Information technology — Coding of audio-visual objects —

Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)

AMENDMENT 1: LASeR extensions

Replace all occurrences of the word "browser" with "LASeR engine".

In Clause 2, add the following references:

ISO/IEC 9899:1990, *Information technology — Programming Languages C*

ISO/IEC 14882, *Programming languages — C++*

ISO/IEC 16262:2002, *Information technology — ECMAScript language specification*

IETF BCP 13, *RFC 4288 on Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*

IETF RFC 3023, XML Media Types, M. Murata, S. St.Laurent, D. Kohn, January 2001, http://www.ietf.org/rfc/rfc3023.txt

IETF RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, January 2005, http://www.ietf.org/rfc/rfc3986.txt

Add the following definition of waiting tree to clause 3:

waiting tree
separate tree defined in addition to the scene tree

NOTE The compositor and renderer have no knowledge of the waiting tree, thus objects in the waiting tree are neither composited nor rendered.

Add the following subclause to Clause 5:

5.2 LASeR Systems Decoder Model

5.2.1 Introduction

The purpose of the LASeR Systems decoder model is to provide an abstract view of the behaviour of the terminal complying with this International Standard. It may be used by the sender to predict how the receiving terminal will behave in terms of buffer management and synchronization when decoding data received in the

form of elementary streams. The LASeR systems decoder model includes a timing model and a buffer model. The LASeR systems decoder model specifies:

1. the conceptual interface for accessing data streams (Delivery Layer),

2. decoding buffers for coded data for each elementary stream,

3. the behavior of elementary stream decoders,

4. composition memory for decoded data from each decoder, and

5. the output behavior of composition memory towards the compositor.

Each elementary stream is attached to one single decoding buffer.

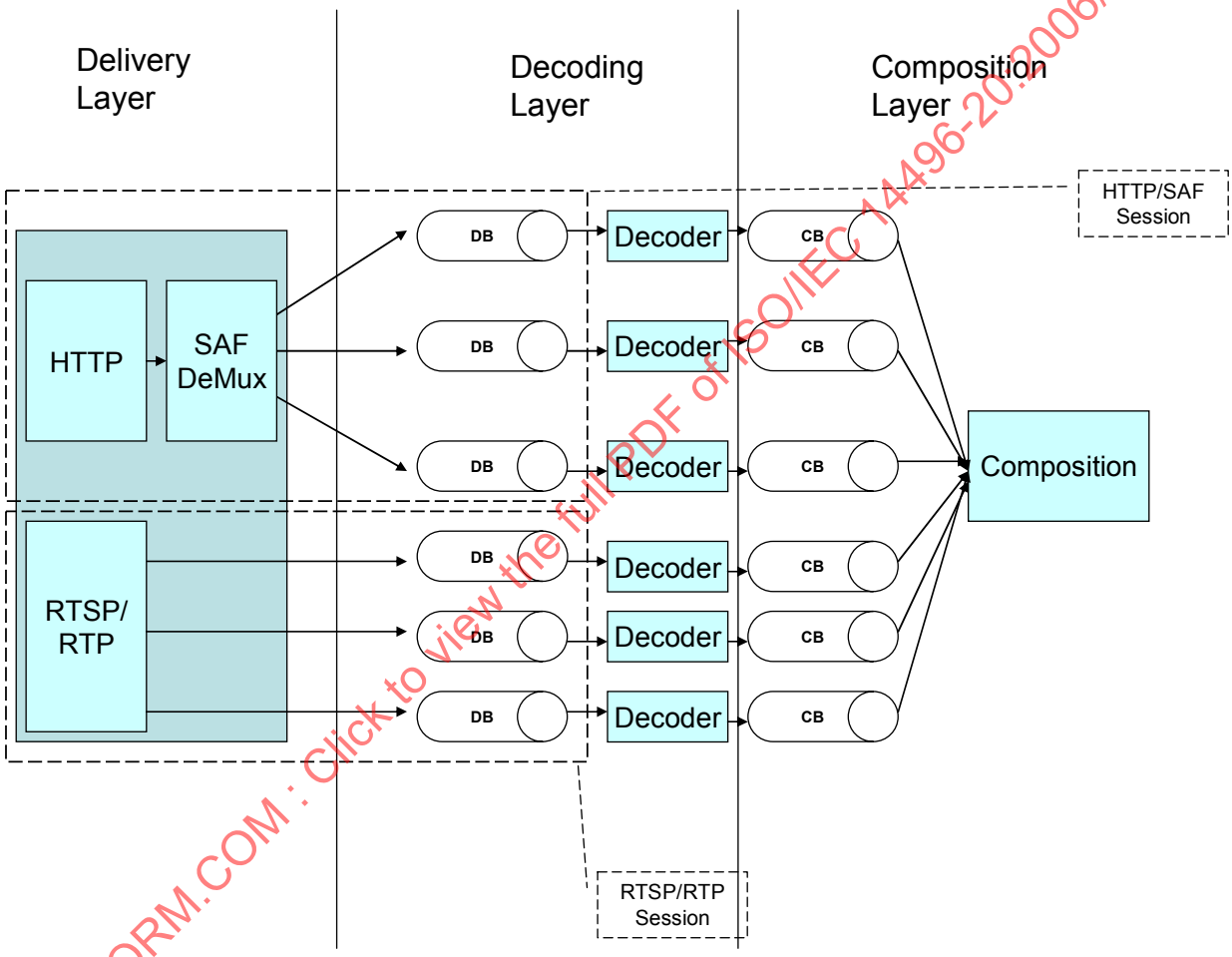

Figure AMD1.1 – LASeR Systems Decoder Model

The definition in ISO/IEC 14496-1 of Access Unit, Decoding Buffer(DB), elementary stream (ES), Decoder (CU) and Composition Unit apply.

5.2.2 Decoder Model

The decoder model as specified in ISO/IEC 14496-1:2004 subclause 7.4.1 applies.

5.2.3 Decoding Buffer

The needed decoding buffer size is known by the sending terminal and conveyed to the receiving terminal as specified in 7.6. The size of the decoding buffer is measured in bytes. The decoding buffer is filled at the rate given by the maximum bit rate for this elementary stream while data is available and with a zero rate otherwise. The maximum bit rate is conveyed by the sending terminal as a part of the decoder configuration information during the set up phase for each elementary stream (see 7.6).

5.2.4 Decoder model with grouped streams

This decoder model may be enhanced when used for group of multiple elementary streams.

In such case, only one composition buffer for the group of streams is used for composition.

When such streams are grouped, and when the setup of multiple decoding chains are available, it is possible, although not mandatory, not to decode all streams at a time.

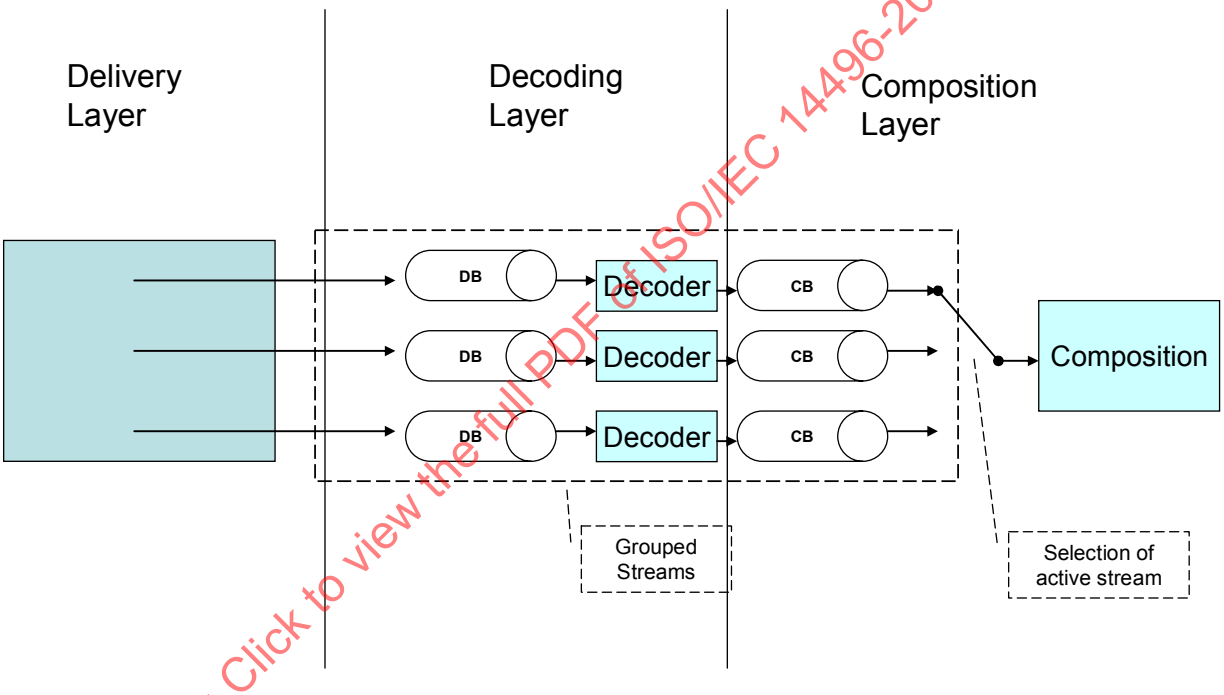

Figure AMD1.2 – Stream grouping with specified System Decoder Model (multiple decoders)

It is indeed expected that multiple decoders may not be available in lightweight terminals or that some delivery scenarios do not allow for having all streams available at the same time (e.g. in broadcast scenarios, the delivery layer could only tune in to one of the streams). The usage of new information about this grouping enables a smart usage of buffers and decoders.

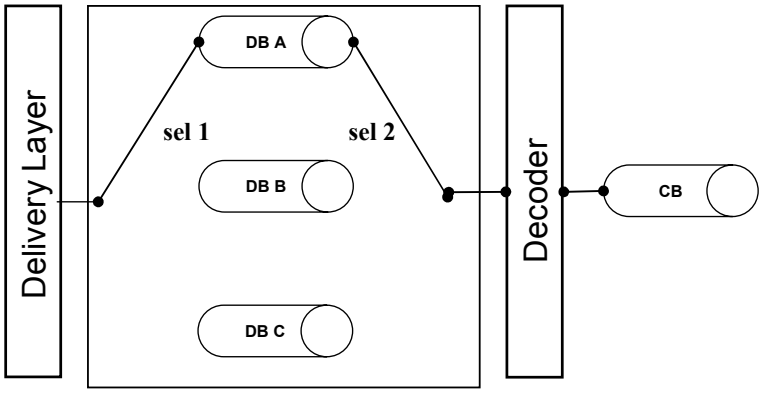

Figure AMD1.3 – Broadcast example of streams grouping, showing a potential optimization using a single decoder.

When only a subset of the group of streams can be accessed at a time (e.g. broadcast scenario depicted above), the selection of the active stream corresponds to a request for the corresponding streams. Nevertheless, the buffer model for stream grouping does not assume immediate reception of data after such request and therefore the active decoding buffer may continue to be used by the decoder up to the moment at which data is available for the newly available stream. In this case the decoding buffer associated to the newly connected stream can be associated with the decoder. At this point the terminal may discard any remaining access units in the previous decoding buffer.

Add the following text to subclause 6.2.1 Scene Tree, at the end:

The API defined in Appendix A of [2] with IDL definitions in Appendix B of the same document can be used to access the LASeR scene tree from programming languages such as ECMA-Script [ISO/IEC 16262], Java [5], C [ISO/IEC 9899:1990] or C++ [ISO/IEC 14882:2002].

Replace subclause 6.5 with:

6.5. Events

6.5.1. Purposes of events

As in SVG, LASeR defines events following the XML Event specification [6]. The events defined in LASeR relate to the management of the network session and decoding chains (including decoding buffers). The events defined in the following subclauses can be used by elements in the scene such as script elements being associated through the listener element, in order to respond to such events.

Note : For instance, in a progressive download scenario, the "buffering" event could be listened by a script in order to trigger a text indicating that content will be played shortly.

```
...
<ev:listener handler="#myscript" event="LASeRBuffering">
<script id="myscript">
   <lsr:Replace ref="#text" attributeName="visibility" value="visible">
</script>
<text id="text" visibility="hidden">Content is being buffered</text>
...
```

In the previous example, the LASeRBuffering event is being listened to by a script "myscript". When the event is launched by the browser, the visibility attribute of the text element is set to "visible".

These events are launched by the LASeR browser either at the "Network" Layer of the browser in which case, the scope of these events is the session or at the decoding chain level, in which case the events are at the stream level.

NOTE a session is identified by a unique url and streams are identified by a streamID.

6.5.2. Events imported from SVG Tiny

The list of supported events with their properties is given in Table AMD1.1.

Table AMD1.2 – List of supported events from SVGT1.2

Event name	Namespace	Description	Bubble	Canc.
“focusin” or “DOMFocusIn”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“focusout” or “DOMFocusOut”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“activate” or “DOMActivate”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“click”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“mousedown”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“mouseup”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“mouseover”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“mouseout”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	Yes
“mousemove”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“load” (or deprecated “SVGLoad”)	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“resize” (or deprecated “SVGResize”)	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“scroll” (or deprecated “SVGScroll”)	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“zoom” (or deprecated “SVGZoom”)	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“beginEvent”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“endEvent”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“repeatEvent”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	Yes	No
“keyup”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“keydown”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“textInput”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“mouseWheel”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“timer”	http://www.w3.org/2001/xml-events	As defined in subclause 13.2 of [2].	No	No
“preload”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“loadProgress”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“postLoad”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“connectionConnected”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“connectionClosed”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“connectionError”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“connectionDataSent”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No
“connectionDataReceived”	http://www.w3.org/2000/svg	As defined in subclause 13.2 of [2].	No	No

6.5.3. Pseudo-events

Pseudo-events are shortcuts created by combinations of other events. Their definition follows:

Table AMD1.3 – List of pseudo-events

Event name	Namespace	Description	Bubble	Canc.
“accessKey(keyCode)”	http://www.w3.org/2000/svg	The key keyCode has been pressed, as defined in subclause 16.2.7 of [2]. This pseudo-event is triggered immediately by a listener on keydown placed on the document node.	No	No
“longAccessKey(keyCode)”	urn:mpeg:mpeg4:laser:2005	This pseudo-event is a combination of a listener on keydown and a listener on keyup placed on the document node; this pseudo-event is triggered if after a system-defined time A after the keydown event, no keyup event has happened.	No	No
“repeatKey(keyCode)”	urn:mpeg:mpeg4:laser:2005	This pseudo-event is a combination of a listener on keydown and a listener on keyup placed on the document node; this pseudo-event is triggered after a system-defined time B after the keydown event, repeatedly every system-defined period C, until a keyup event happens.	No	No
“shortAccessKey(keyCode)”	urn:mpeg:mpeg4:laser:2005	This pseudo-event is a combination of a listener on keydown and a listener on keyup placed on the document node; this pseudo-event is triggered if a keyup event happens after a time shorter than A after the keydown event. This is exclusive of longAccessKey(k).	No	No

6.5.4. LASeR Events

LASeR defines the following events:

Table AMD1.4 – List of LASeR events

Event name	Namespace	Description	Bubble	Canc.
“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
“play”	urn:mpeg:mpeg4:laser:2005	Starts or resumes the clock of the timed object they are sent to, and have no effect on non timed objects.	No	No
“pausedEvent”	urn:mpeg:mpeg4:laser:2005	Occurs when a Timed Element is paused	Yes	No
“resumedEvent”	urn:mpeg:mpeg4:laser:2005	Occurs when a Timed Element is resumed	Yes	No
“activatedEvent”	urn:mpeg:mpeg4:laser:2005	Occurs when an element is activated, either as the result of an Activate command or a change in an active attribute to the value true, applied to it or to one of its ancestors.	No	No
“deactivatedEvent”	urn:mpeg:mpeg4:laser:2005	Occurs when an element is deactivated, either as the result of a Deactivate command or a change in an active attribute to the value false, applied to it or to one of its ancestors.	No	No
“screenOrientation0”	urn:mpeg:mpeg4:laser:2005	The screen orientation has changed to typical ‘landscape’ orientation (LASeREvent)	No	No
“screenOrientation90”	urn:mpeg:mpeg4:laser:2005	The screen orientation has changed to typical ‘portrait’ orientation (LASeREvent)	No	No
“screenOrientation180”	urn:mpeg:mpeg4:laser:2005	The screen orientation has changed to inverted ‘landscape’ orientation (LASeREvent)	No	No
“screenOrientation270”	urn:mpeg:mpeg4:laser:2005	The screen orientation has changed to inverted ‘portrait’ orientation (LASeREvent)	No	No
“stop”	urn:mpeg:mpeg4:laser:2005	Upon receiving such an event, a timed element behaves as if the uDOM method endElement() was called. (LASeREvent)	No	No

Example:

Typical usage for screen orientation events is to have a animation triggered by one of these events, the animation changing the position/rotation of a group of scene elements to match the new screen orientation:

```
<animateTransform  begin="lsr:screenOrientation90"  to="…"  xlink:href="#object1"
dur="1s"/>

<animateTransform  begin="urn:mpeg:mpeg4:laser:2005:screenOrientation90"  to="…"
xlink:href="#object2" dur="1s"/>

<animateTransform  begin="urn:mpeg:mpeg4:laser:2005:screenOrientation90"  to="…"
xlink:href="#object3" dur="1s"/>

…

<animateTransform  begin="urn:mpeg:mpeg4:laser:2005:screenOrientation0"  to="…"
xlink:href="#object1" dur="1s"/>

<animateTransform  begin="urn:mpeg:mpeg4:laser:2005:screenOrientation0"  to="…"
xlink:href="#object2" dur="1s"/>

<animateTransform  begin="urn:mpeg:mpeg4:laser:2005:screenOrientation0"  to="…"
xlink:href="#object3" dur="1s"/>

…
```

6.5.5. General IDL definition of LASeR events.

```
interface LASeREvent : events::Event {};
```

All LASeR specific events are prefixed with LASeR.

6.5.6. IDL Events definitions

6.5.6.1 ExternalValueEvent

```
interface ExternalValueEvent : LASeREvent {
       readonly attribute float absoluteValue;
       readonly attribute boolean computableAsFraction;
       readonly attribute float fraction;
};
```

No defined constants

Attributes

- **absoluteValue**: This value represent the status of a resource of any kind, e.g. the remaining battery time.
- **computableAsFraction:** This value indicates whether a fraction can be computed from the absoluteValue.
- **fraction:** This value shall be between 0 and 1 inclusively and represent the status of the resource, e.g. the fraction of remaining battery time over operation time when fully charged.

No defined methods

Example: The following event could be defined:

"batteryState"	urn:example:X	Received by the document at system-dependent intervals and informs about the fraction of battery charge remaining. (ExternalValueEvent)	No	No

and used as follows:

```
<svg viewBox="0 0 1000 1000" baseProfile="tiny" id="root"
   xmlns="http://www.w3.org/2000/svg" version="1.1"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:X="urn:example:X">
   <rect width="100" height="100" stroke="blue"
         stroke-width="1">
     <animateColor id="myBatteryAnim" attributeName="fill"
                   begin="indefinite" from="red" to="blue"
                   dur="indefinite">
  </rect>
   <ev:listener observer="root" event="X:batteryState"
                handler="#myBatteryAnim">
</svg>
```

Add the following text to 6.7.1 before "The following restrictions apply to all commands":

- ***Activate***: to transfer an element from the waiting tree to the scene tree
- ***Deactivate***: to transfer an element from the scene tree to the waiting tree
- ***ReleaseResource***: to instruct the LASeR engine that a resource (typically media stream) will no longer be used in the scene and may therefore be reclaimed.

Add the following text at the end of subclause 6.7.1:

A separate DOM tree is defined in addition to the scene tree: the waiting tree. The compositor and renderer have no knowledge of the waiting tree, thus objects in the waiting tree are neither composited nor rendered. The LASeR Commands are extended to search for elements first in the scene tree, then in the waiting tree if the elements are not found in the scene tree. The commands Activate and Deactivate are defined to operate on the waiting tree.

NOTE Elements placed in the waiting tree can be updated but are not accessible from the scene tree. A use element pointing to an element placed into the waiting tree behaves as if the referred element did not exist, i.e. nothing is rendered. Deactivating / activating a subtree with listeners may imply unregistering / registering the listeners observing elements not in the waiting tree.

Example:

```
<!-- Module 1 -->
<g id="module1">
      <rect x="0" y="10" width="176" height="20" fill="green"/>
      <text x="88" y="25" text-anchor="middle">Press FIRE</text>
      <ev:listener event="accessKey(FIRE)" handler="#go1"/>
      <ev:listener event="foo.click" handler="#go3"/>
      <animate xlink:href="#foo" …/>
   ...
</g>

<!-- Module 2 -->
<g id="module2">
      <rect x="0" y="10" width="176" height="20" fill="lime"/>
      <text x="88" y="25" text-anchor="middle">Press FIRE</text>
      <ev:listener event="accessKey(FIRE)" handler="#go2"/>
      <ev:listener event="foo.click" handler="#go4"/>
      <animate xlink:href="#foo" …/>
   ...
</g>
```

```
<g id=”foo”>
   ...
</g>

<!-- Modules aggregation and control -->
<lsr:conditional begin="0;accessKey(1)">
      <lsr:Deactivate ref="#module2"/>
      <lsr:Activate ref="#module1"/>
</lsr:conditional>
<lsr:conditional begin="accessKey(2)">
      <lsr:Deactivate ref="#module1"/>
      <lsr:Activate ref="#module2"/>
</lsr:conditional>
```

This simple example shows 2 independent modules. Since they use both the FIRE key, they cannot be both “active” at the same time. The two conditionals at the end implement very simply an exclusive activation of the two modules.

Add the following text at end of 6.7.10.1 (Save Command) Semantics:

Only attributes of elements identified by string IDs can be saved and restored reliably across scenes.

Attributes referencing elements by ID can only be saved and restored reliably across scenes if they hold a string ID.

In subclause 6.7.11.2, add a bullet:

- ***intvalue2***: this attribute defines the second int value of the event.

In subclause 6.7.11.2, replace the table with:

Name of event	Attribute of event	Attribute of SendEvent
TimeEvent	detail	intvalue
UIEvent	detail	intvalue
WheelEvent	wheelDelta	intvalue
MouseEvent	screenX	*not carried*
	screenY	*not carried*
	clientX	pointvalue
	clientY	pointvalue
	button	intvalue
TextEvent	data	stringvalue
KeyEvent	keyIdentifier	when used with accessKey, longAccessKey, shortAccessKey and repeatKey, the key identifier follows the event name, e.g. accessKey(LEFT); with other key events, use stringvalue when unknown keyIdentifier, otherwise use intvalue with LASeR-defined keyIdentifier
ProgressEvent	lengthComputable	*not carried, the LASeR engine shall set this value to true if the value total is present*
	loaded	intvalue
	total	invalue2

After subclause 6.7.11, add the following subclauses:

6.7.12 Activate

6.7.12.1 Semantics

The Activate command reverses the effect of the Deactivate command, i.e. it restores the target element from the waiting tree to the scene tree. From the DOM point of view, the effect on the element is that the lsr:ghost element is replaced by the element, e.g. using a DOM replaceChild.
Note – The lsr:ghost element may have been transferred to the waiting tree as part of a Activate command applied to one of its ancestors.

6.7.12.2 Attributes

- ***ref***: the id of the element which shall be restored from the waiting tree to the scene tree.

6.7.13 Deactivate

6.7.13.1 Semantics

The Deactivate command takes an element and places it into the waiting tree, so that it can be restored later to the scene tree by a Activate command. From the DOM point of view, the effect on the element is:

- the element is replaced by an lsr:ghost element, e.g. using a DOM replaceChild, regardless of whether the element is in the scene tree or in the waiting tree. Note: the element could already be in the waiting tree if one of its ancestors has been placed in the waiting tree.
- the element is placed in the waiting tree.

NOTE The lsr:ghost element is never encoded nor transmitted nor rendered.

6.7.13.2 Attributes

- ***ref***: the id of the element which shall be placed in the waiting tree.

6.7.14 ReleaseResource

6.7.14.1 Semantics

The ReleaseResource command instructs the LASeR engine that the indicated resource will no longer be used in the scene. All associated LASeR engine resources (memory, I/O) may be reclaimed. All scene elements that may be referencing this resource behave as if they pointed at a non available resource.

6.7.14.2 Attributes

- ***ref***: this attribute defines the resource which may be discarded. It may be a stream ID or any other URI.

In subclause 6.8.1, replace sentence:

The list of possible attributes for an element is given in the summary table in subclause 6.8.53.

with:

The list of possible attributes for an element is given in the summary table in subclause 6.9.

Add to subclause 6.8.4.1:

When an animate element is not active and it receives an ExternalValueEvent, then the element sets the animation target attribute value as if the animation progress was at the fraction contained in the ExternalValueEvent.

When an animate element is not active and it receives a ProgressEvent which has lengthComputable=true, then the element sets the animation target attribute value as if the animation progress was at the fraction loaded over total.

Add to subclause 6.8.5.1:

When an animateColor element is not active and it receives an ExternalValueEvent, then the element sets the animation target attribute value as if the animation progress was at the fraction contained in the ExternalValueEvent.

When an animateColor element is not active and it receives a ProgressEvent which has lengthComputable=true, then the element sets the animation target attribute value as if the animation progress was at the fraction loaded over total.

Add to subclause 6.8.6.1:

When an animateMotion element is not active and it receives an ExternalValueEvent, then the element sets the animation target attribute value as if the animation progress was at the fraction contained in the ExternalValueEvent.

When an animateMotion element is not active and it receives a ProgressEvent which has lengthComputable=true, then the element sets the animation target attribute value as if the animation progress was at the fraction loaded over total.

Add to subclause 6.8.7.1:

When an animateTransform element is not active and it receives an ExternalValueEvent, then the element sets the animation target attribute value as if the animation progress was at the fraction contained in the ExternalValueEvent.

When an animateTransform element is not active and it receives a ProgressEvent which has lengthComputable=true, then the element sets the animation target attribute value as if the animation progress was at the fraction loaded over total.

Add to subclause 6.8.10.1:

The children of the conditional element are not accessible through DOM or LASeR Commands.

Add to subclause 6.8.28.2:

- **width**: the width of the clipping rectangle. This can also be accessed through the x component of the size field. This attribute is animatable but not inheritable.
- **height**: the height of the clipping rectangle. This can also be accessed through the y component of the size field. This attribute is animatable but not inheritable.
- **x**: the horizontal coordinate of the center of the clipping rectangle. This attribute is animatable but not inheritable.
- **y**: the vertical coordinate of the center of the clipping rectangle. This attribute is animatable but not inheritable.

At the end of subclause 6.8.30.1 Semantics (of LASeR selector), add the following text:

In the following, N is the number of children of the selector element. The *choice* attribute determines the actual rendering mode:

1. *choice* >= 0 & *choice* < N: only the child of index *choice* is displayed, i.e. is "on" while the other children are "off". The "off" children are neither composed nor rendered. animate* elements and conditionals in the tree below "off" children are inactive.
2. *choice* == none | *choice* >= N: nothing is displayed. All children are "off", i.e. neither composed nor rendered. animate* elements and conditionals in the tree below "off" children are inactive.
3. *in all other cases*: all the children are displayed at (0,0) of the local coordinate system without clipping.

In case 1. and 2., when a child changes from "off" to "on", its children with begin attributes get activated; when a child changes from "on" to "off", its children with end attributes get stopped.

At the end of subclause 6.8.34.2 Attributes (of svg), add the following text:

- **lsr:fullscreen:** the attribute 'lsr:fullscreen' is defined on the <svg> element to hint that the LASeR scene should be rendered on the entire screen. The attribute can take two values "true" or "false", with false (normal rendering) being the default value. With the attribute set to true the LASeR UE should negotiate the rendering area with its parent UE and get as large part of the screen as possible for the LASeR rendering area. The attribute is neither animatable nor inheritable.

At the end of subclause 6.8.35 SVGT11 switch, add the following text:

This specification defines four extension strings in order to allow the use of the switch element to control the initial layout of a scene according to the screen orientation:

- orient0 for typical 'landscape' orientation
- orient90 for typical 'portrait' orientation
- orient180
- orient270

An example of usage is:

```
<switch>
    <g requiredExtensions="orient90">
       … layout for portrait …
    </g>
    <g requiredExtensions="orient0">
       … layout for landscape…
    </g>
</switch>
```

After subclause 6.8.40, add the following subclauses:

6.8.41 AnimateScroll

6.8.41.1 Semantics

The purpose of this element is to allow to:

- scroll a set of objects of unknown size, inside a clipping rectangle.
- define the speed of the scrolling in terms of clip size, not in terms of the objects size
- define the bounds of scrolling precisely, according to the actual size of the objects

- define the beginning and ending conditions (screen full or empty of the objects)
- define automatic and manual scrolling (continuous scrolling, page advance on action…), and allow a combination of automatic and manual
- define scroll stops (for manual scrolling) by page (size of viewport) or by object markers

The animateScroll element is designed in a manner similar to SMIL/SVG animate* elements. It works in combination with a clipping region (lsr:rectClip element). Scrolling can be automatic or manual (e.g. triggered by key events). The scrolling effect is obtained by a translation applied to all children of the lsr:rectClip element. Multiple animateScroll and setScroll elements acting on the same objects combine in the same way as multiple animateTransform elements with additive=”sum”.

Each frame where it is active, animateScroll does:

- if the element or one of its attributes has changed, recompute the element size
- depending on the parameters, compute the new position of the scrolled objects

The direction attribute of animateScroll is not sensitive to rotation.

Example 1: Plain vanilla

```
<lsr:rectClip id="t1" size="120 20">
  <text…>Que j’aime à faire apprendre ce nombre utile au sage. Immortel
Archimède, toi de qui Syracuse garde encore la mémoire.</text>
</lsr:rectClip>
<animateScroll xlink:href="#t1" direction="left" speed="2"
               delayAtStart="1" delayAtEnd="1" begin="3"
               repeatDur="indefinite"/>
```

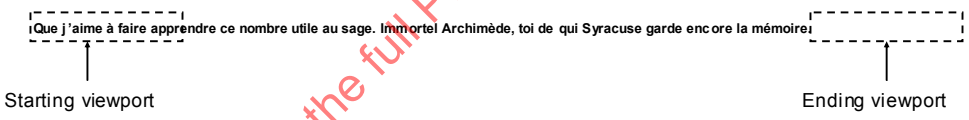

Figure AMD1.4 – Start and end states of Example 1

Example 2: Start and end conditions

```
<lsr:rectClip id="t1" size="120 20">
  <text…>Que j’aime à faire apprendre ce nombre utile au sage. Immortel
Archimède, toi de qui Syracuse garde encore la mémoire.</text>
</lsr:rectClip>
<animateScroll xlink:href="#t1" direction="left" speed="2"
   delayAtStart="1" delayAtEnd="1" begin="3"
   repeatDur="indefinite" from="100 0" to="100 0"/>
```

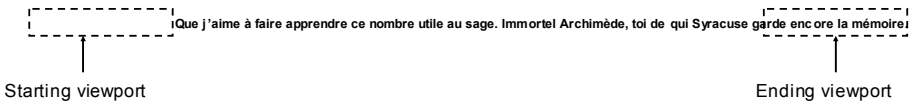

Figure AMD1.5 – Start and end states of Example 2

The units of the from and to attributes are percents of the clip size. When the direction is “left”, from=”100 0” means that the object is initially 100% of clip with to the right of the clip, which means right outside of the clip. to=”100 0” means that the scrolled objects are, at the end of the animation, one clip width further to the left than the default leading edge of the clipping zone in the scrolling direction.

Example 3: Manual mode

```
<lsr:rectClip id="t1" size="120 20">
  <text…>Que j'aime à faire apprendre ce nombre utile au sage. <tspan
id="t2"/>Immortel Archimède, toi de qui Syracuse garde encore la mémoire.</text>
</lsr:rectClip>
<setScroll xlink:href="#t1" increment="100 0" begin="accessKey(RIGHT)"/>
<setScroll xlink:href="#t1" increment="-100 0" begin="accessKey(LEFT)"/>
<setScroll xlink:href="#t1" to="#t2" begin="accessKey(2)"/>
```

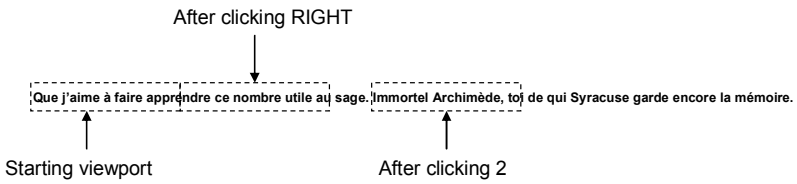

Figure AMD1.6 – Start, interactive and end states of Example 3

Example 4: Mixed mode

```
<lsr:rectClip id="t1" size="120 20">
  <text…>Que j'aime à faire apprendre ce nombre utile au sage. <tspan
id="t2">Immortel Archimède,</tspan> toi de qui Syracuse garde encore la
mémoire.</text>
</lsr:rectClip>
<animateScroll xlink:href="#t1" begin="0;accessKey(0)" repeatDur="indefinite"
               direction="left" speed="3" delayAtStart="1" delayAtEnd="1"
               end="accessKey(2);accessKey(RIGHT);accessKey(LEFT)"/>
<setScroll xlink:href="#t1" increment="100 0" begin="accessKey(RIGHT)"/>
<setScroll xlink:href="#t1" increment="-100 0" begin="accessKey(LEFT)"/>
<setScroll xlink:href="#t1" to="#t2" begin="accessKey(2)"/>
```

The above sample has one scrolling to the left in 3 seconds per clip width with 1s waiting at each end, and looping indefinitely, unless the user presses a key. Pressing RIGHT or LEFT move the text by one page (one clip width) left or right. Pressing 2 goes to "Immortel". Pressing 0 resumes the automatic scrolling.

6.8.41.2 Attributes

- xlink:href: this attribute specifies the ID of an lsr:rectClip element containing the scrolled objects. In the absence of this attribute, this element does not have any effect.
  Animatable: no. Inheritable: no.
- by: this attribute specifies the relative translation and consists in 2 values, expressed in percents of the clip size. The default value is 0 0. In case of repeated scrolling, this functions as an animateTransform with accumulate="sum".
  Animatable: no. Inheritable: no.
- from: this attribute specifies the starting position for the scrolling and consists in either 2 values, expressed in percents of the clip size, or in an element ID. The default value is 0 0, for the leading edge of the clipping zone in the scrolling direction. When from is an ID, it shall be the ID of a child of the scrolled object; the scrolling shall start with the leading edge of the object matching the leading edge of the clipping rectangle, according to the scrolling direction. from defines the minimum position of the clip.
  Animatable: no. Inheritable: no.
- to: this attribute specifies the end of scrolling position and consists in either 2 values, expressed in percents of the clip size, or in an element ID. The default value is 0 0, for the leading edge of the clipping zone in the scrolling direction. When to is an ID, it shall be the ID of a child of the scrolled object; the scrolling shall end with the trailing edge of the object matching the trailing edge of the clipping rectangle,

according to the scrolling direction. to defines the maximum position of the clip.
Animatable: no. Inheritable: no.

- delayAtStart: this attribute specifies the delay between the start of a cycle and the beginning of the movement. The default value is 0 for no delay.
Animatable: no. Inheritable: no.
- delayAtEnd: this attribute specifies the delay between the end of the movement and the end of the cycle (and the restart of the cycle if the animation is looping). The default value is 0 for no delay.
Animatable: no. Inheritable: no.
- direction: this enumeration specifies the scrolling direction: up, down, left, right. For example, “up” means that the automatic scrolling move the scrolled objects toward the negative Ys. The default value is left.
Animatable: no. Inheritable: no.
- speed: this attribute expresses the time to scroll one page (or size of clipping zone) of text. It is exclusive from dur/end. The default value is 0 for no scrolling at all.
Animatable: no. Inheritable: no.
- begin: this attribute specifies the beginning of the animation, in a similar manner as for the other animate* elements. The default value is indefinite.
Animatable: no. Inheritable: no.
- dur: this attribute specifies the duration of the animation, including possible values of delayAtStart and delayAtEnd. The value of dur, if specified, shall be greater than delayAtStart+delayAtEnd. The default value is 0 for no scrolling.
Animatable: no. Inheritable: no.
- end: the difference between the value of begin and end specifies the duration of the animation, including possible values of delayAtStart and delayAtEnd. The value of end, if specified, shall be greater than begin+delayAtStart+delayAtEnd. The default value is indefinite.
Animatable: no. Inheritable: no.
- restart: this attribute specifies the restartability of the animation, in a similar manner as for the other animate* elements. The default value is always.
Animatable: no. Inheritable: no.
- repeatCount: this attribute specifies the repeatability of the animation, in a similar manner as for the other animate* elements. The default value is 0 for no repetition.
Animatable: no. Inheritable: no.
- repeatDur: this attribute specifies the repeatability of the animation, in a similar manner as for the other animate* elements. The default value is 0 for no repetition.
Animatable: no. Inheritable: no.
- fill: this attribute specifies whether the text returns to its initial position before the animation (remove) or keeps its last position (freeze). The default value is freeze.
Animatable: no. Inheritable: no.

6.8.42 SMIL animation

6.8.42.1 Semantics

The ***SMIL animation*** element is specified in subclause 7.3.1 of [W3C SMIL2]. Neither local IDs nor global IDs are shared between the main scene containing the animation element and the subscene referred by the animation element.

6.8.42.2 Attributes

- **clipBegin**: this attribute is defined in subclause 7.5.1 of [SMIL2]. The value represents a normal play time. The value of clipBegin is taken into account when the media element is activated. The play time of some streams cannot be controlled, and under these circumstances, this attribute has no effect. This attribute is not animatable and not inheritable.

- **clipEnd**: this attribute is defined in subclause 7.5.1 of [SMIL2]. The value represents a normal play time. The play time of some streams cannot be controlled, and under these circumstances, this attribute has no effect. This attribute is not animatable and not inheritable.
- **syncReference**: this attribute holds a reference to the stream whose clock acts as a clock reference for the stream referred to by this element. This attribute is not animatable and not inheritable.
- **syncBehavior**: this attribute is defined in subclause 6.3.1 of [SMIL2]. This attribute is not animatable and not inheritable.
- **syncTolerance**: this attribute is defined in subclause 6.3.1 of [SMIL2]. This attribute is not animatable and not inheritable.

6.8.43 SVGT1.1 font

The ***SVGT1.1 font*** element is specified in subclause 20.3 of [W3C SVG11]

6.8.44 SVGT1.1 font-face

The ***SVGT1.1 font-face*** element is specified in subclause 20.8.3 of [W3C SVG11]

6.8.45 SVGT1.1 font-face-src

The ***SVGT1.1 font-face-src*** element is specified in subclause 20.8.3 of [W3C SVG11]

6.8.46 SVGT1.1 font-face-uri

The ***SVGT1.1 font-face-uri*** element is specified in subclause 20.8.3 of [W3C SVG11]

6.8.47 SVGT1.1 font-face-name

The ***SVGT1.1 font-face-name*** element is specified in subclause 20.8.3 of [W3C SVG11]

6.8.48 SVGT1.1 glyph

The ***SVGT1.1 glyph*** element is specified in subclause 20.4 of [W3C SVG11]

6.8.49 SVGT1.1 missing-glyph

The ***SVGT1.1 missing-glyph*** element is specified in subclause 20.5 of [W3C SVG11]

6.8.50 SVGT1.1 hkern

The ***SVGT1.1 hkern*** element is specified in subclause 20.7 of [W3C SVG11]

6.8.51 LASeR setScroll

6.8.51.1 Semantics

The setScroll element is a simplified animateScroll. Examples are provided in the animateScroll subclause.

6.8.51.2 Attributes

- xlink:href: this attribute specifies the ID of an lsr:rectClip element containing the scrolled objects. In the absence of this attribute, this element does not have any effect.
  Animatable: no. Inheritable: no.

- increment: this attribute specifies the relative translation and consists in 2 values, expressed in percents of the clip size. The scrolling increment is applied as many times as the element has been activated. The default value is 0 0.
  Animatable: no. Inheritable: no.
- to: this attribute specifies an absolute scrolling position and consists in either 2 values, expressed in percents of the clip size, or in an element ID. The default value is 0 0, for the leading edge of the clipping zone in the scrolling direction. When to is an ID, it shall be the ID of a child of the scrolled object; the scrolled objects shall be placed such that the leading edge of the scrolled objects match the leading edge of the clipping rectangle, according to the scrolling direction.
  Animatable: no. Inheritable: no.
- direction: this enumeration specifies the scrolling direction: up, down, left, right. For example, “up” means that the automatic scrolling moves the scrolled objects toward the negative Ys. The default value is left.
  Animatable: no. Inheritable: no.
- begin: this attribute specifies the beginning of the animation, in a similar manner as for the other animate* elements. The default value is indefinite.
  Animatable: no. Inheritable: no.

6.8.52 LASeR StreamSource

6.8.52.1 Semantics

The purpose of StreamSource is to first make available at the scene level information about the state of media chains (e.g. buffering, availability for rendering …) and secondly to give hints for potential resource optimization like, for instance, usage of a single hardware decoder, smooth transition during rendering without silence or black screens, ads display during channel switching.

The ‘streamSource’ element manages a set of streams, it does not actually have any display/rendering functionality. Instead, it keeps available an up-to-date pixel or audio buffer that can be used, and thus displayed by one or more video elements. As such, streamSource makes a pixel/audio buffer available to video/audio elements.

If any of the elements (in the source attribute) in the streamSource element is an audiovisual element containing both audio and video, two buffers (for audio and video respectively) shall be made available for all elements referring to the streamSource.

In such case,

- the pixel buffer associated to the streamSource when an audio only stream is active will represent a black buffer, the size of which will be defined by the width and height elements. Children of the streamSource element may define the default buffer in that case.
- the audio buffer associated to the streamSource when an video only stream is active will render as silence. Children of the streamSource element may define a default audio clip in that case.

The 'streamSource' element is able to trigger events such as the availability of the desired stream.

6.8.52.2 Attributes

- sources : is an array of references to stream sources
- sourceIndex : is the current active video source. On loading of the scene or when changing the sourceIndex, the browser shall connect to the session defined by the url corresponding to sourceIndex, this is source[i]
- width, height : default pixel buffer size
- mode : is one of "replace", "useOld", "keepOld", "playList".

- The value "replace" indicates that the composition buffer associated to the stream should be immediately replaced when sourceIndex is changed.
- The value "useOld" means that the previous stream will be decoded till the stream pointed by sourceIndex is ready which means that data will start to be available for composition for the stream pointed by sourceIndex. This implies that decoding and rendering of the previous stream will occur up to the moment at which the first composition buffer is made available.
- With "keepOld"[1] the same behaviour as from "useOld" is expected, furthermore the previous video stream source is kept open.
- The value "playList" indicates that the different sources will be played in sequence. In this case, it is possible to append a "repeat" and/or "shuffle" modes such as "playList, repeat, shuffle". This modes repeat and shuffle allow respectively to cycle through sources and to shuffle the sources.

6.8.53 LASeR updates

6.8.53.1 Semantics

The LASeR updates element provides a link to a source of scene updates. The value of the xlink:href attribute defines the location of the updates to be applied to the current scene. The updates element is a timed element providing stream playback control.

6.8.53.2 Attributes

All SMIL timing attributes defined in [2] subclause 16.2.7 are defined for this element, except the "fill" attribute.

- **clipBegin**: this attribute is defined in subclause 7.5.1 of [SMIL2]. The value represents a normal play time. The value of clipBegin is taken into account when the media element is activated. The play time of some streams cannot be controlled, and under these circumstances, this attribute has no effect. This attribute is not animatable and not inheritable.
- **clipEnd**: this attribute is defined in subclause 7.5.1 of [SMIL2]. The value represents a normal play time. The play time of some streams cannot be controlled, and under these circumstances, this attribute has no effect. This attribute is not animatable and not inheritable.
- **xlink:href**: this attribute specifies the location of the stream of updates. In the absence of this attribute, this element does not have any effect. This attribute is not animatable and not inheritable.
- **syncReference**: this attribute holds a reference to the stream whose clock acts as a clock reference for the stream referred to by this element. This attribute is not animatable and not inheritable.
- **security:** this attribute controls the type of allowed updates in the referenced stream. The referenced stream may contain a complete scene (for example, LASeR starting with a NewScene command) or a scene segment (for example, LASeR starting with any other command than NewScene or RefreshScene). Allowed values for the security attribute are *any, complete* or *segment.* The value *any* does not restrict the type of updates in the referenced stream: this is the default value. The value *complete* restricts the updates in the referenced stream to be a complete scene: the current scene may be discarded upon activation of the hyperlink. The value *segment* restricts the updates in the referenced stream to be a scene segment: any command replacing or refreshing the entire scene (for example LASeR NewScene/RefreshScene) shall be ignored. This attribute is not animatable and not inheritable.
- **flow:** this attribute controls the behavior of the referenced stream of updates. Allowed values are *parent* and *new*. The name of the attribute refers to the “flow of information”, which comprises incoming commands and media. The value *parent* means that the referenced stream shall replace the flow of information from which this a element originated. The value *new* means that the referenced stream shall be processed in parallel with existing flows of information. Note – the LASeR engine may manage transport connections according to this information. This attribute is not animatable and not inheritable.

[1] This richer functionality allows for coming back to the previous channel but would suppose to keep two streams open (i.e. in a DVB-H case to syntonise two bursts).

Renumber subclause 6.8.41 Summary of Possible Attributes per Element to 6.9.

In subclause 6.9 Summary of Possible Attributes per Element, replace the table 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 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 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
lsr:animateScroll	id class xml:base xml:lang xml:space xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show by from to delayAtStart delayAtEnd speed direction begin dur end fill restart repeatCount repeatDur
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
animation	id class xml:base xml:lang xml:space requiredFeatures requiredExtensions systemLanguage requiredFormats requiredFonts audio-level display image-rendering pointer-events shape-rendering text-rendering viewport-fill viewport-fill-opacity visibility lsr:rotation lsr:scale lsr:translation transform xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show 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 fill focushiglight width height x y externalResourcesRequired begin end dur min max restart repeatCount repeatDur syncBehavior syncTolerance syncMaster pereserveAspectRatio type lsr:syncReference lsr:clipBegin lsr:clipEnd initialVisibility
audio	audio-level begin class dur end externalResourcesRequired id lsr: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 lsr:rotation lsr:scale lsr: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
lsr:conditional	id class begin enabled externalResourcesRequired xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show xml:base xml:lang xml:space
lsr:cursorManager	id class 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 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 xml:base xml:lang xml:space
font	id class xml:base xml:lang xml:space externalResourcesRequired horiz-adv-x horiz-origin-x
font-face	id class xml:base xml:lang xml:space accent-height alphabetic ascent bbox cap-height descent externalResourcesRequired font-family font-stretch font-style font-variant font-weight hanging ideographic mathematical overline-position overline-thickness panose-1 slope stemh stemv strikethrough-position strikethrough-thickness underline-position underline-thickness unicode-range units-per-em widths x-height
font-face-src	id class xml:base xml:lang xml:space
font-face-uri	id class xml:base xml:lang xml:space externalResourcesRequired xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show
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 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
glyph	id class xml:base xml:lang xml:space arabic-form d glyph-name horiz-adv-x lang unicode externalResourceRequired xlink:href xlink:title xlink:type xlink:role xlink:arcrole xlink:actuate xlink:show
hkern	id class xml:base xml:lang xml:space g1 g2 k u1 u2

Element name	Attributes
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 lsr:rotation lsr:scale lsr: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 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 x1 x2 xml:base xml:lang xml:space y1 y2
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
missing-glyph	id class xml:base xml:lang xml:space d horiz-adv x
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

Element name	Attributes
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
lsr:rectClip	id class 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 width height x y
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 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:setScroll	id class xml:base xml:lang xml:space begin increment to direction xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type
lsr:simpleLayout	id class 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

Element name	Attributes
streamSource	id class xml:base xml:lang xml:space sources sourceIndex width height mode
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
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
updates	externalResourcesRequired requiredExtensions requiredFeatures requiredFormats systemLanguage xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type begin class dur end id lsr:syncReference repeatCount repeatDur syncBehavior syncTolerance type xml:base xml:lang xml:space clipBegin clipEnd security flow
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

After subclause 6.9, add the following subclause:

6.10 Addition to the uDOM API

6.10.1 Parsing of an encoded XML element

6.10.1.1 Syntax

```
Node parseLASeRBinaryElement(in sequence<octet> data, in Document contextDoc);
```

6.10.1.2 Semantics

Given a chunk of binary data and a Document object, parse the binary data as a LASeR element and return a Node representing it. If the binary data is not well-formed, this method must return a `null` value.

The format of the sequence<octet> data parameter is: one LASeRHeader followed by a byte-aligned encoded object of class updatable_elements as defined in clause 12.

6.10.2 Trait access extensions

The following table adds to the table in A.8.12 of [SVGT1.2]. It contains trait access rules for DIMS extensions.

Attribute	Trait Getter	Trait Setter	Default Values	Description
fullscreen	getTraitNS[true | false]	setTraitNS[true | false]	false	Available on <video> and <svg> elements
x	getFloatTraitNS	setFloatTraitNS	0.0f	Origin x of the <rectClip>
y	getFloatTraitNS	setFloatTraitNS	0.0f	Origin y of the <rectClip>
width	getFloatTraitNS	setFloatTraitNS	0.0f	Width of the clipping region defined by <rectClip>
height	getFloatTraitNS	setFloatTraitNS	0.0f	Height of the clipping region defined by <rectClip>

6.10.3 Description of getFloatTraitNS and setFloatTraitNS methods

float getFloatTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);

- Same as getFloatTrait, but for namespaced traits. Parameter name must be a non-qualified trait name, i.e. without prefix.

Parameters:

`namespaceURI` - the namespaceURI of the trait to retrieve.

`name` - the name of the trait to retrieve.

Return Value:

the trait value as float.

Exceptions:

`DOMException` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.

`DOMException` - with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a float.

void setFloatTraitNS(in DOMString namespaceURI, in DOMString name, in float value) raises(DOMException);

Same as setFloatTrait, but for namespaced traits. Parameter name must be a non-qualified trait name, i.e. without prefix.

Parameters:

`namespaceURI` - the namespaceURI of the trait to be set.
`name` - the name of the trait to be set.
`value` - the value of the trait to be set as float.

Exceptions:

`DOMException` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
`DOMException` - with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as a float (for e.g. NaN)
`DOMException` - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null.

In clause 7, replace Table 8 with:

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	GroupDescriptor	-
0x09	FirstFragmentUnit	The first Fragment of an Access Unit
0x0A	FragmentUnit	A Fragment of an Access Unit (not the first fragment)
0x0B	SAFConfiguration	A safConfiguration object
0x0C	StopCache	-
0x0D ~ 0x0F	Reserved	-

At the end of 7.11 EndOfSAFSession, add the following text:

When a Saf Session is closed (either by *EndOfSAFSession*, or by other means), all streams that have a String streamID shall not be discarded.

After subclause 7.12, add the following subclauses:

7.13 GroupingDescriptor

7.13.1 Syntax

```
class groupingDescriptor {
    bit(8) number_of_elements;
    for (int i=0 ; i<number_of_element ; i++) {
        bit(12) streamID;
     }
}
```

7.13.2 Semantics

number_of_elements – number of elements in this group

streamID – a streamID of elementary stream grouped into a single decoder

This descriptor signals grouping of elementary streams to use more than one decoder buffers for a single decoder to improve switching behaviours between the elementary streams in the same group. The terminal shall continuously receive elementary streams referred by this descriptor and fill the decoding buffers. No more than one stream referred by this descriptor shall be decoded at the same time. This may require allocation of more than one decoding buffer for a single decoder. Note: When this descriptor is used, even though the decoder buffers are not connected to the decoder, the AU is removed from the decoder buffer when the DTS is arrived.

7.14 SAF Fragment Unit

The fragment unit is a specialized SAF Extended Access Unit. accessUnitLength of this packet shall be zero.

7.14.1 Syntax

```
class safFU {
bit(4) accessUnitType;
bit(12) streamID;
bit(16) payloadLength;
bit(8) fragmentSeqNum;
byte(8)[payloadLength-1] payload;
}
```

7.14.2 Semantics

`accessUnitType` – the value of this field shall be 10

`streamID` – same as in 7.5.2.

`payloadLength` – the length of the data part of the access unit.

`fragmentSeqNum` – the fragment sequence number on 8 bits, possibly wrapping.

`payload` – the data part of the access unit. The size of the payload is signalled by the `payloadLength` field in the packet header as specified in 7.4.

7.15 SAF First Fragment Unit

The first fragment unit is a specialized SAF Extended Access Unit. accessUnitLength of this packet must be zero.

7.15.1 Syntax

```
class safFFU {
  bit(4) accessUnitType;
  bit(12) streamID;
  bit(16) payloadLength;
  bit(4) carriedAccessUnitType;
  bit(4) reserved;
  bit(32) totalLengthOfAccessUnit;
  byte(8)[payloadLength-5] payload;
}
```

7.15.2 Semantics

`accessUnitType` – the value of this field shall be 9

`streamID` – the reference of the media stream this AU belongs to.

`carriedAccessUnitType` – an indication about the type of the access unit whose fragment is carried in the payload. Detailed values of `accessUnitType` and the data corresponding to each type are defined in Table 8.

`totalLengthOfAccessUnit` – the length of the access unit, which shall be equal to the accumulated length of the fragments.

`payload` – the data part of the access unit. The size of the payload is signalled by the `payloadLength` field in the packet header as specified in 7.4.

7.16 SAF Configuration

7.16.1 Syntax

```
class safConfiguration {
  bit(1) add;
  bit(1) cacheThisSession;
  bit(1) hasMainStreamID;
  bit(1) hasStreamDependencies;
  if (add == 0 && hasMainStreamID == 1) {
    bit(12) mainStreamID;
  } else {
    bit(4) reserved = 0;
  }
  if (hasStreamDependencies) {
    uint(16) sd_count;
    for (int i = 0; i < sd_count; i++) {
      bit(12) streamID;
      bit(12) dependsOnStreamID;
    }
  }
  // globalStreamTable: external or internal global streamID references
  uint(8) count;
  for (int i = 0; i < count; i++) {
```

```
    bit(12) localStreamIdForThisGlobal[[i]];
      bit(12) globalNameLength;
    byte[globalNameLength] globalName[[i]];
  }
  if (cacheThisSession == 1) {
      bit(1) permanent;
      bit(31) validity;
      unit(16) urlLength;
      byte(urlLength) cacheURL;
  }
  byte[accessUnitLength - sizeof(previous elements)] extensionData;
}
```

7.16.2 Semantics

`add` – if set to 1, this packet defines an addition to the previous SAF Configuration. If set to 0, the current global stream table is reset and the global stream table defined in this packet becomes the current global stream table. Existing stream dependencies shall not be re-assined, either when appending a new configuration or when redefining it.

`cacheThisSession` – if set, the current session should be cached. If the connection is closed before the arrival of a stopCache instruction, nothing is stored in cache.

`permanent` – if true, the cached session shall be kept, if the terminal has enough resources, after the end of the application for a duration stored in the validity field.

`validity` – the validity period of this cached session expressed in seconds

`urlLength` – if non-zero, the cacheURL field contains a string of length urlLengthwhich is the URL at which the current session should be cached. If zero, the original URL of the current session should be used for caching. Note: the presence of a URL should be the exception.

`cacheURL` – the URL to be used for the caching of this session.

`mainStreamID` – in cases where there are multiple scene streams, the mainstreamed allows the author to specify which is the main stream. In the absence of a SAF Configuration packet, the main stream ID is 0.

`streamID` – stream ID for which a coding dependency is being specified.

`dependsOnStreamID` – ID of the stream upon which the specified stream is dependent.

`localStreamIdForThisGlobal` – defines the number used as stream ID for the stream whose name is globalName.

`globalName` – defines the name of the global stream, which is the identifier used for reference across SAF session boundaries.

7.17 Stop Cache

7.17.1 Syntax

```
class stopCache {
}
```

7.17.2 Semantics

When this packet arrives, any caching of the current session ends.

At the end of subclause 8, add this subclause:

8.4. LASeR Core

8.4.1. Applications

Rich-media services on mid- and lower-end embedded devices.

The purpose of LASeR Core is to allow J2ME implementations of LASeR. One specific feature of Core 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.

8.4.2. List of Tools/Functionalities

8.4.2.1. video element

The attribute **transformBehavior** shall be restricted to values “pinned | pinned90 | pinned180 | pinned270”.

The attribute **overlay** shall be restricted to values “top”.

8.4.2.2. stroking

The attributes **stroke-linecap** and **stroke-linejoin** are restricted to the value “butt” and “miter” respectively.

The attributes **stroke-miterlimit, stroke-dasharray** and **stroke-dashoffset** are forbidden.

8.4.2.3. inheritance

Attributes with a possible value of “inherit” are only allowed on objects which use their value directly, not on objects that pass the value to their children. The “inherit” value is forbidden.

Hints such as *-rendering are only allowed on the root svg.

pointer-events is only allowed on the root svg.

level 1: pointer-events is limited to values bounding-box and none.

8.4.2.4. animation

For elements **animate**, **animateColor**, **animateTransform** and **animateMotion**, the following restrictions apply:

- the value of the attribute **fill** is restricted to “freeze”,
- the value of the attribute **additive** is restricted to “replace”,
- the value of **calcMode** is restricted to “linear” or “discrete”, but on animateMotion,
- either attribute **from** or **values** must be specified.

The value of restart is restricted to "***always***".

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

8.4.2.5. xlink and xml

Attributes xml:base, xml:lang, xml:space, xlink:title, xlink:type, xlink:role, xlink:arcrole, xlink:actuate and xlink:show are forbidden.

8.4.2.6. LASeR Commands

color-rendering, shape-rendering, text-rendering, image-rendering, attributeName are not updatable.

Update of attributes from timed elements are restricted in the following manner:

- accumulate, additive, attributeName, by, calcMode, dur, fill, from, id, keyPoints, keySplines, keyTimes, max, min, path, repeatCount, repeatDur, restart, to, values, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base cannot be updated
- begin, end, clipBegin and clipEnd may be updated

8.4.2.7. uDOM

uDOM is not mandatory in LASeR Core profile

8.4.2.8. Navigation

The events focusin and focusout are not supported. The attribute focusable, focusHighlight and nav-* are not supported.

8.4.2.9. text

The attributes x and y of text can only take single values, not lists of coordinates.

The attribute xml:space is mandatory with value "preserve" on text and tspan.

The possible children for text are restricted to: tspan.

8.4.2.10. Units

Only % and px are allowed as length units in LASeR Core.

8.4.2.11. transform

The ref() construct is not allowed in transform attribute values.

8.4.2.12. System Paint Servers

System paint servers are not allowed.

8.4.2.13. Events

The following events are not allowed: DOMFocusIn, DOMFocusOut, click, keydown, keyup, resize, scroll, zoom, rotate, timer, connection*.

8.4.2.14. SMIL hyperlinking

Hyperlinks to an element, whether timed or not, are not allowed.

8.4.2.15. use

The xlink:href attribute of the use element is restricted to designating the ID of an element of the current scene. More complex URLs are not part of LASeR Core.

8.4.3. Comparison with existing profiles and object types

LASeR Core is a superset of LASeR mini

8.4.4. Profile definition

This table defines the allowed elements in the profile and the list of their possible attributes, possibly with restrictions when listed in bold.

Element name	Attributes
a	id display transform xlink:href
animate	id lsr:enabled begin end dur repeatCount repeatDur **restart="always" attributeName fill="freeze"** to **xlink:href(no replace)** from by values **calcMode="discrete|linear"** keyTimes **additive="replace"** accumulate
animateColor	id lsr:enabled begin end dur repeatCount repeatDur **restart="always" attributeName fill="freeze"**to **xlink:href(no replace)** from by values **calcMode="discrete|linear"**keyTimes **additive="replace"** accumulate
animateMotion	id path keyPoints rotate lsr:enabled begin end dur repeatCount repeatDur **restart="always" fill="freeze"** to **xlink:href(no replace)** from by values **calcMode="discrete|linear|paced"** keyTimes **additive="replace"** accumulate
lsr:animateScroll	id xml:base **xlink:href(no replace)** by **from(no id) to(no id)** delayAtStart delayAtEnd speed direction begin dur end **fill="freeze"** restart repeatCount repeatDur
animateTransform	id type lsr:enabled begin end dur repeatCount repeatDur **restart="always" attributeName fill="freeze"**to **xlink:href(no replace)** from by values **calcMode="discrete|linear"**keyTimes **additive="replace"** accumulate
audio	id begin end clipBegin clipEnd repeatCount repeatDur audio-level xlink:href
circle	id cx cy r display fill fill-opacity fill-rule stroke stroke-opacity stroke-width transform vector-effect
conditional	id begin type lsr:enabled
lsr: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 transform vector-effect
foreignObject	id width height x y viewport-fill viewport-fill-opacity display
g	id display transform
image	id width height x y **transformBehavior(no geometric)** display transform xlink:href
line	id x1 y1 x2 y2 display stroke stroke-opacity stroke-width transform vector-effect
ev:listener	id lsr:enabled event observer handler propagate defaultAction target
metadata	id
mpath	id xlink:href
path	id d display fill fill-opacity fill-rule stroke stroke-opacity stroke-width transform vector-effect
polygon	id points display fill fill-opacity fill-rule stroke stroke-opacity stroke-width transform vector-effect
polyline	id points display fill fill-opacity fill-rule stroke stroke-opacity stroke-width transform vector-effect
rect	id width height x y rx ry display fill fill-opacity fill-rule stroke stroke-opacity stroke-width transform vector-effect
lsr:rectClip	id size display transform
lsr:selector	id choice display transform
set	id lsr:enabled begin end dur repeatCount repeatDur **restart="always" attributeName fill="freeze"** to **xlink:href(no replace)**

lsr:setScroll	id begin increment to direction **xlink:href(no replace)** xml:base
lsr:simpleLayout	id delta display transform
svg	id width height viewBox preserveAspectRatio **zoomAndPan="disable"** viewport-fill viewport-fill-opacity display pointer-events color-rendering shape-rendering image-rendering text-rendering
text	id display fill fill-opacity fill-rule font-family font-size font-style font-weight stroke stroke-opacity stroke-width text-anchor editable x y transform text-decoration **xml:space="preserve"**
title	id
tspan	id display fill fill-opacity fill-rule font-family font-size font-style font-weight stroke stroke-opacity stroke-width text-decoration **xml:space="preserve"**
use	id overflow x y display transform xlink:href
lsr:updates	begin clipBegin clipEnd end flow xlink:href id repeatCount repeatDur security lsr:syncReference
video	id begin display end clipBegin clipEnd repeatCount repeatDur audio-level xlink:href width height x y **overlay="top"** lsr:fullscreen **transformBehavior(no geometric)** transform

8.4.5. Level definitions

Level 0: only one scene stream active at a time

Add attached files as electronic attachments and add the following text at the end of clause 11:

- LASeRML/saf2.xsd: (normative) documents the syntax of an XML equivalent to the SAF binary syntax for use in LASeR conformance and reference software activities

- LASeRML/laser-datatypes2.xsd: (normative) documents all datatypes used in the other schemas

- LASeRML/laser2.xsd: (normative) documents the XML syntax of LASeR Commands for use in LASeR conformance and reference software activities

- LASeRML/svg2.xsd: (normative) documents the XML syntax of the common part between SVG and LASeR; it is not intended to validate SVGT documents; this is for use in LASeR conformance and reference software activities

- sdl_amd1.txt: (normative) documented in the next subclause

In subclause 12.1.3, replace table 10 with:

codePoint	**privateDataContainer**	**extConfiguration**
	defines encoding	*defines configuration for*
0	anyXML as defined in 12.2.3.20	not used
1	opaque binary data	opaque binary data
2	ISO/IEC 23001-1 encoding	ISO/IEC 23001-1 compliant decoderInit
3-7	reserved for ISO use	reserved for ISO use
8-15	to be defined by RA	to be defined by RA

Add the following text to subclause 12.1.3:

If codePoint 2 is used, then the data is ISO/IEC 23001-1 compliant private extensions. Decoders not compliant with the signalled ISO/IEC 23001-1 private extension (identified by the namespace) are not mandated to decode this private extension data and may skip it.

The following sentences ONLY apply for decoders compliant with both ISO/IEC 23001-1 and ISO/IEC 14496-20 :

If no decoderInit signalled in LASeRHeader, then the decoderInit is defined in subclause 13.5. If decoderInit is signalled then:

- decoderInit is a compatible extension of the one defined in subclause 13.5.
- the extensionIDBits of LASeRHeader correspond to the number of bits that shall be used for coding the total number of schemas declared in the decoderInit.
- The attached 'encoding' schemas shall be used for the configuration of ISO/IEC 23001-1 (encoding schemas are normative).
- The Classification scheme and type codec association described in subclause 13.3 shall be used(classificiation schemes and association are normative)
- The transformation process between the validation schema and the encoding schemas are described informatively in clause 13.

Replace subclause 12.2 with:

12.2 Binary Syntax

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) codePoint;
  switch(codePoint){
    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 occ1;
    for(int t=0;t<occ1;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_0to1float 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:
      update_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) codePoint;
  switch(codePoint){
```

```
    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 occ1;
    for(int t=0;t<occ1;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;
```

```
  }
}
// AMD1 elements
// to be enclosed in an element_any with extensionID = 2

class SVGAmd1Extension {
  vluimsbf5 occ0;
  for(int t=0;t<occ0+1;t++) {
    bit(4) ch5;
    switch(ch5){
    case 0:
      element_animation animation;
      break;
    case 1:
      element_discard discard;
      break;
    case 2:
      element_font font;
      break;
    case 3:
        element_fontFace font_face;
      break;
    case 4:
        element_fontFaceSrc font_face_src;
      break;
    case 5:
        element_fontFaceUri font_face_uri;
      break;
    case 6:
      element_glyph glyph;
      break;
    case 7:
      element_handler handler;
      break;
    case 8:
        element_hkern element_hkern;
      break;
    case 9:
        element_missingGlyph missing_glyph;
      break;
    case 10:
        element_prefetch element_prefetch;
      break;
    case 11:
      element_solidColor solidColor;
      break;
    case 12:
      element_tBreak tBreak;
      break;
    case 13:
      element_textArea textArea;
      break;
    default:
      break;
    }
  }
}
class element_animation {
  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_fill;
  if (has_fill) {
    attr_animFill fill;
  }
  bit(1) has_height;
  if (has_height) {
    attr_custom_coordinate height;
  }
  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_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_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_discard {
  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_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_font {
  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_horiz_adv_x;
  if(has_horiz_adv_x) {
```

```
    attr_custom_fixed_16_8 horiz_adv_x;
  }
  bit(1) has_horiz_origin_x;
  if(has_horiz_origin_x) {
    attr_custom_fixed_16_8 horiz_origin_x;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_fontFace {
  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_accent_height;
  if(has_accent_height) {
    attr_custom_fixed_16_8 accent_height;
  }
  bit(1) has_alphabetic;
  if (has_alphabetic) {
    attr_custom_fixed_16_8 alphabetic;
  }
  bit(1) has_ascent;
  if (has_ascent) {
    attr_custom_fixed_16_8 ascent;
  }
  bit(1) has_bbox;
  if (has_bbox) {
    attr_custom_byteAlignedString bbox;
  }
  bit(1) has_cap_height;
  if(has_cap_height) {
    attr_custom_fixed_16_8 cap_height;
  }
  bit(1) has_descent;
  if (has_descent) {
    attr_custom_fixed_16_8 descent;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_font_family;
  if(has_font_family) {
    attr_custom_byteAlignedString font_family;
  }
  bit(1) has_font_stretch;
  if(has_font_stretch) {
    attr_custom_byteAlignedString font_stretch;
  }
  bit(1) has_font_style;
  if(has_font_style) {
    attr_custom_byteAlignedString font_style;
  }
  bit(1) has_font_variant;
  if(has_font_variant) {
```

```
    attr_custom_byteAlignedString font_variant;
  }
  bit(1) has_font_weight;
  if(has_font_weight) {
    attr_custom_byteAlignedString font_weight;
  }
  bit(1) has_hanging;
  if (has_hanging) {
    attr_custom_fixed_16_8 hanging;
  }
  bit(1) has_ideographic;
  if (has_ideographic) {
    attr_custom_fixed_16_8 ideographic;
  }
  bit(1) has_mathematical;
  if (has_mathematical) {
    attr_custom_fixed_16_8 mathematical;
  }
  bit(1) has_overline_position;
  if(has_overline_position) {
    attr_custom_fixed_16_8 overline_position;
  }
  bit(1) has_overline_thickness;
  if(has_overline_thickness) {
    attr_custom_fixed_16_8 overline_thickness;
  }
  bit(1) has_panose_1;
  if(has_panose_1) {
    attr_custom_byteAlignedString panose_1;
  }
  bit(1) has_slope;
  if (has_slope) {
    attr_custom_fixed_16_8 slope;
  }
  bit(1) has_stemh;
  if (has_stemh) {
    attr_custom_fixed_16_8 stemh;
  }
  bit(1) has_stemv;
  if (has_stemv) {
    attr_custom_fixed_16_8 stemv;
  }
  bit(1) has_strikethrough_position;
  if(has_strikethrough_position) {
    attr_custom_fixed_16_8 strikethrough_position;
  }
  bit(1) has_strikethrough_thickness;
  if(has_strikethrough_thickness) {
    attr_custom_fixed_16_8 strikethrough_thickness;
  }
  bit(1) has_underline_position;
  if(has_underline_position) {
    attr_custom_fixed_16_8 underline_position;
  }
  bit(1) has_underline_thickness;
  if(has_underline_thickness) {
    attr_custom_fixed_16_8 underline_thickness;
  }
  bit(1) has_unicode_range;
  if(has_unicode_range) {
```

```
    attr_custom_byteAlignedString unicode_range;
  }
  bit(1) has_units_per_em;
  if(has_units_per_em) {
    attr_custom_fixed_16_8 units_per_em;
  }
  bit(1) has_widths;
  if (has_widths) {
    attr_custom_byteAlignedString widths;
  }
  bit(1) has_x_height;
  if(has_x_height) {
    attr_custom_coordinate x_height;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_fontFaceSrc {
  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 child1;
}
class element_fontFaceUri {
  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_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_glyph {
  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_arabic_form;
  if(has_arabic_form) {
    attr_custom_byteAlignedString arabic_form;
  }
  bit(1) has_d;
  if (has_d) {
    attr_custom_path d;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_glyph_name;
  if(has_glyph_name) {
    attr_custom_byteAlignedString glyph_name;
  }
  bit(1) has_horiz_adv_x;
  if(has_horiz_adv_x) {
    attr_custom_fixed_16_8 horiz_adv_x;
  }
  bit(1) has_lang;
  if (has_lang) {
    attr_custom_byteAlignedString lang;
  }
  bit(1) has_unicode;
  if (has_unicode) {
    attr_custom_byteAlignedString unicode;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_handler {
  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_event;
  if (has_event) {
    attr_custom_event event;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_hkern {
```

```
  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_g1;
  if (has_g1) {
    attr_custom_byteAlignedString g1;
  }
  bit(1) has_g2;
  if (has_g2) {
    attr_custom_byteAlignedString g2;
  }
  bit(1) has_k;
  if (has_k) {
    attr_custom_fixed_16_8 k;
  }
  bit(1) has_u1;
  if (has_u1) {
    attr_custom_byteAlignedString u1;
  }
  bit(1) has_u2;
  if (has_u2) {
    attr_custom_byteAlignedString u2;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_missingGlyph {
  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_d;
  if (has_d) {
    attr_custom_path d;
  }
  bit(1) has_horiz_adv_x;
  if(has_horiz_adv_x) {
    attr_custom_fixed_16_8 horiz_adv_x;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_prefetch {
  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_bandwidth;
  if (has_bandwidth) {
    vluimsbf5 bandwidth;
  }
  bit(1) has_mediaCharacterEncoding;
  if (has_mediaCharacterEncoding) {
    attr_custom_byteAlignedString mediaCharacterEncoding;
  }
  bit(1) has_mediaContentEncodings;
  if (has_mediaContentEncodings) {
    attr_custom_byteAlignedString mediaContentEncodings;
  }
  bit(1) has_mediaSize;
  if (has_mediaSize) {
    vluimsbf5 mediaSize;
  }
  bit(1) has_mediaTime;
  if (has_mediaTime) {
    vluimsbf5 mediaTime;
  }
  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_solidColor {
  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_tBreak {
```

```
  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 child1;
}
class element_textArea {
  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_editable;
  if(has_editable) {
    attr_editable editable;
  }
  bit(1) has_height;
  if (has_height) {
    attr_custom_coordinate height;
  }
  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_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class update_Activate {
  attr_custom_IDREF ref;
  bit(1) has_attr_any;
```

```
  if (has_attr_any) {
    attr_any any;
  }
}
class update_Deactivate {
  attr_custom_IDREF ref;
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
}
class update_ReleaseResource {
  attr_custom_anyURI ref;
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
}
class LASeRAmd1Extension {
  vluimsbf5 occ1;
  for(int t=0;t<occ1+1;t++) {
    bit(2) ch5;
    switch(ch5){
      case 0:
        element_animateScroll animateScroll;
        break;
      case 1:
        element_setScroll setScroll;
        break;
      case 2:
        element_streamSource streamSource;
        break;
      case 3:
        element_updates updates;
        break;
      }
  }
}
class element_animateScroll {
  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_by;
  if (has_by) {
    attr_scrollStop by;
  }
  bit(1) has_delayAtEnd;
  if (has_delayAtEnd) {
    attr_time delayAtEnd;
  }
  bit(1) has_delayAtStart;
```

```
  if (has_delayAtStart) {
    attr_time delayAtStart;
  }
  bit(1) has_direction;
  if (has_direction) {
    attr_direction direction;
  }
  bit(1) has_dur;
  if (has_dur) {
    attr_time dur;
  }
  bit(1) has_fill;
  if (has_fill) {
    attr_animFill fill;
  }
  bit(1) has_from;
  if (has_from) {
    attr_scrollStop from;
  }
  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_speed;
  if (has_speed) {
    attr_time speed;
  }
  bit(1) has_to;
  if (has_to) {
    attr_scrollStop to;
  }
  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_setScroll {
  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_direction;
  if (has_direction) {
    attr_direction direction;
  }
  bit(1) has_increment;
  if (has_increment) {
    attr_scrollStop increment;
  bit(1) has_to;
  if (has_to) {
    attr_scrollStop to;
  }
  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_streamSource {
  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_height;
  if (has_height) {
    attr_custom_coordinate height;
  }
  bit(1) has_mode;
  if (has_mode) {
    attr_streamSourceMode mode;
  }
  bit(1) has_sourceIndex;
  if (has_sourceIndex) {
    vluimsbf5 sourceIndex;
  }
  attr_sources sources;
  bit(1) has_width;
  if (has_width) {
    attr_custom_coordinate width;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}
class element_updates {
  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) 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_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_flow;
if(has_flow) {
  // Enumeration: any{0} complete{1} segment{2}
  bit(2) flow;
}
bit(1) has_syncReference;
if(has_syncReference) {
  attr_custom_anyURI syncReference;
}
bit(1) has_security;
if(has_security) {
  // Enumeration: new{0} parent{1}
  bit(2)security;
}
bit(1) has_attr_any;
if (has_attr_any) {
```

```
    attr_any any;
  }
  object_content child0;
}
class LASeRAmd1Command {
  vluimsbf5 occ3;
  for(int t=0;t<occ3+1;t++) {
    bit(2) ch5;
    switch(ch5){
    case 0:
        privateElementContainer privateElementContainer; // to allow anyXML at the updates level (top
level of the AU)
      break;
    case 1:
        update_Activate Activate;
      break;
    case 2:
        update_Deactivate Deactivate;
        break;
    case 3:
        update_ReleaseResource ReleaseResource;
        break;
   }
  }
}
//*************************************************************************************************
// start specific classes
//*************************************************************************************************

class LASeRUnit {
  LASeRUnitHeader h;
  initialisations c;
  vluimsbf5 occ1;
  for(int t=0;t<occ1+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
class element_any {
  uint(extensionIDBits) extensionID;
  vluimsbf5 len; //length in bits
  if (extensionID == 2) { //LASeR AMD1
    const vluimsbf5 reserved = 87; // or "const bit(10) reserved = 599;"
    vluimsbf5 occ2;
    for(int t=0;t<occ2+1;t++) {
      bit(2) ch5;
      switch(ch5){
        case 0:
          SVGAmd1Extension SVGAmd1Extension;
          break;
        case 2:
          LASeRAmd1Extension LASeRAmd1Extension;
          break;
      }
    }
  } else {
    bit[len] toSkip;
  }
}

class update_any {
  uint(extensionIDBits) extensionID;
  vluimsbf5 len; //length in bits
  if (extensionID == 2) { //LASeR AMD1
    const vluimsbf5 reserved = 87; // or "const bit(10) reserved = 599;"
    vluimsbf5 occ2;
    for(int t=0;t<occ2+1;t++) {
      const bit(2) reserved = 1;
      LASeRAmd1Command LASeRAmd1Command;
    }
  } else {
    bit[len] toSkip;
  }
}

//for attributes
class attr_any {
  do {
    uint(extensionIDBits) extensionID;
    vluimsbf5 len; //length in bits
    if (extensionID == 2) { //LASeR AMD1
      bit(1) hasRequiredFonts;
      if (hasRequiredFonts) {
        attr_custom_byteAlignedString requiredFonts;
      }
      bit(1) hasX;
      if (hasX) {
        attr_custom_coordinate x;
      }
      bit(1) hasY;
      if (hasY) {
        attr_custom_coordinate y;
      }
    } else {
      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) hasExtConfiguration;
  if (hasExtConfiguration) {
    vluimsbf5 len;
    byte[len] extConfiguration; //extConfiguration is defined in Table 10
  }
  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;
    // globalStreamTable: external global streamID references
    vluimsbf5 countGS;
    for (int i = 0; i < count; i++) {
      vluimsbf5 localStreamIdForThisGlobal[[i]];
      byteAlignedStringClass globalName[[i]];
    }
    bit[len2] remainingData;
    // len2 is defined implicitly as: len - sizeof(globalStreamTable)
  }
}
```

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]];
  }
}
class attr_scrollStop {
  bit(1) choice;
  switch(choice) {
    case 0:
      attr_custom_IDREF IDREF0;
      break;
    case 1:
      int max=+2;
      for (int t=0;t<max;t++) {
        attr_custom_fixed_16_8 item[[t]];
      }
      break;
  }
}
class attr_direction {
  // Enumeration: down{0} left{1} right{2} up{3}
  bit(4) direction;
}
class attr_streamSourceMode {
  // Enumeration: keepOld{0} playList{1} replace{2} useOld{3}
    bit(3) streamSourceMode;
}
class attr_sources {
  vluimsbf5 max;
  for (int t=0;t<max;t++) {
    attr_custom_anyURI item[[t]];
  }
}
```

12.2.3 Specific Data Types

12.2.3.1 ID

12.2.3.1.1 Syntax

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

12.2.3.1.2 Semantics

This class allocates a number for an id.

12.2.3.2 IDRef

12.2.3.2.1 Syntax

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

12.2.3.2.2 Semantics

This class allows pointing to the integer value defined by the ID_class.

12.2.3.3 AnyURI

12.2.3.3.1 Syntax

```
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;
    }
}
```

12.2.3.3.2 Semantics

This class allows the encoding of three forms of URI usable in LASeR: string, stream ID and element ID.

12.2.3.4 Color

12.2.3.4.1 Syntax

```
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;
      }
  }
}
```

12.2.3.4.2 Semantics

This class allows pointing to a color. The `colorIndexBits` value shall be initialised using the initialisation parameters as defined in subclause 6.6.2.2. The value for each color shall be initialised using the colorInitialisation class in the LASeRUnit, whose syntax is given below in subclause 12.2.

12.2.3.5 Matrix

12.2.3.5.1 Syntax

```
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;
      }
    }
}
```

12.2.3.5.2 Semantics

This class decodes a matrix value. The `coordBits` and `scaleBits` values shall be initialised using the initialisation parameters as defined in subclause 6.6.2.2.

12.2.3.6 Fraction

12.2.3.6.1 Syntax

```
class attr_custom_0to1float {
    uint(8) quantifiedValue; // uniform quantization of a float between 0 and 1
}
```

12.2.3.6.2 Semantics

This class decodes an opacity value. This class is a `UniformQuantizer` where v_{min} = 0 v_{max} = 1 and nbits = 8 (see UniformQuantizer advanced optimised decoder in [ISO/IEC 23001-1].)

12.2.3.7 Path

12.2.3.7.1 Syntax

```
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]];
    }
}
```

12.2.3.7.2 Semantics

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.

12.2.3.8 PointSequence

12.2.3.8.1 Syntax

```
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 ;
}
```

12.2.3.8.2 Semantics of the SDL elements

- flag - Flag indicating FL encoding (flag = 0) or EG encoding (flag = 1).
- kvalue - Parameter for EG encoding that varies according to geometric distribution. For example, as kvalue increases, the slope of the geometric distribution becomes gentler.
- XMvalue, YMvalue - The number of leading zeros.
- CodeNum - Code number.
- dx - Differential value between x-coordinate values of a current point and a previous point.

dx = x[i] – x[i – 1]

- dy - Differential value between y-coordinate values of the current point and the previous point.

dy = y[i] – y[i – 1]

- INFO - Value having information about dx or dy..

12.2.3.8.3 Decoding Process

When the LASeR binary stream is assumed to be decoded into a point sequence of (x_0, y_0), (x_1, y_1), …, (x_n, y_n),

- The number of points in the point sequence is extracted from the LASeR binary stream
- The flag is extracted from the LASeR binary stream
- If the value of the flag is zero which indicates the point sequence is encoded using the fixed length coding, decoding according to the following process

- If the number of points is two or less:

    - Each value of x_0, y_0, x_1 and y_1 is extracted by reading “bits” number of bits

- Otherwise:

    (i) Each value of x_0 and y_0 is extracted by reading “bits” number of bits

    (ii) The number “bitsx” of the bits required for a differential value dx of x-coordinates and the number “bitsy” of the bits required for a differential value dy of y-coordinates are extracted

    (iii) dx and dy are extracted by reading “bitsx” bits and “bitsy” bits, respectively, and then $x_i = x_{i-1}$ + dx and $y_i = y_{i-1}$ + dy are calculated

    (iv) i is incremented and the previous step (iii) is performed (n-1) times.

- If the value of the flag is one which indicates the point sequence is encoded using the Exp-Golomb coding, decoding according to the following process

    - The number of points of the point sequence is extracted from the LASeR binary stream

    - The parameter k is extracted from the LASeR binary stream

    - “bits” number of bits are read and then the first point coordinates (x0, y0) are decoded

    - For each point except the point (x_0, y_0), the following process is performed to decode one point (x_i, y_i).

    (i) Bits are read one at a time until “1” is detected and the total number of read bits is set to M

    (ii) The read “1” is discarded

    (iii) (M+k) bits are read and assigned to INFO

    (iv) CodeNum = 2^{M+k} + INFO – 2^k is calculated

    (v) dx is calculated from CodeNum

    (vi) $x_i = x_{i-1}$ + dx is calculated

    (vii) Bits are read one by one until “1” is detected and the total number of read bits is set to M

    (viii) The read “1” is discarded

    (ix) (M+k) bits are read and assigned to INFO

    (x) CodeNum = 2^{M+k} + INFO – 2^k is calculated

    (xi) dy is calculated from CodeNum

    (xii) $y_i = y_{i-1}$ + dy is calculated.

12.2.3.8.4 Encoding Process

This subclause is informative. When a point sequence is assumed to be comprised of (n+1) number of points: (x_0, y_0), (x_1, y_1), …, (x_n, y_n),

- Choose coding method between Exp-Golomb coding or fixed length coding
- When fixed length coding is chosen, encoding the point sequence according to the following process

    - If the number of points is two or less:

    (i) The minimum number of bits with which all of x_0, y_0, x_1, and y_1 can be encoded is calculated and encoded

    (ii) Points (x_0, y_0) and (x_1, y_1) are encoded using the number of bits calculated above

- Otherwise:

  (i) The minimum number of bits with which the point (x_0, y_0) can be encoded is calculated and encoded

  (ii) Point (x_0, y_0) is encoded using the number of bits calculated above

  (iii) dx_{10}, …, dx_{nn-1} (here, $dx_{nn-1} = x_n - x_{n-1}$) are calculated and then the number, bitsx, of bits required for encoding them is calculated

  (iv) dy_{10}, …, dy_{nn-1} (here, $dy_{nn-1} = y_n - y_{n-1}$) are calculated and then the number, bitsy, of bits required for encoding them is calculated

  (v) The numbers of bits, bitsx and bitsy are encoded

  (vi) dx_{10}, dy_{10}, …, dx_{nn-1}, dy_{nn-1} are encoded

- When Exp-Golomb coding is chosen, encoding the point sequence according to the following process

  - The minimum number of bits with which the point (x_0, y_0) can be encoded is calculated and encoded

  - Point (x_0, y_0) is encoded using the minimum number of bits

  - For each point except the point (x_0, y_0), the following process is performed to encode one point (x_i, y_i).

    (i) A difference, “diffx,” between x_i and x_{i-1} is mapped into an EG code number CodeNum without a sign, according to the rule given below:

    If (diffx >= 0) CodeNum = diffx * 2 – 1

    else CodeNum = |diffx| * 2

    (ii) M denoting the number of leading zeros is calculated by $M = | \log_2(CodeNum + 2^k)| - k$

    (iii) M number of “0” bits are recorded.

    (iv) One “1” bit is recorded.

    (v) The suffix offset “INFO,” which carries information, is calculated by $INFO = CodeNum + 2^k - 2^{M+k}$

    (vi) INFO is recorded in (M+k) bits

    (vii) A difference “diffy” between yi and yi-1 is mapped into an EG code number CodeNum

    (viii) Without a sign, according to the rule:

    If (diffy >= 0) CodeNum = diffy * 2 – 1; and

    else CodeNum = |diffy| * 2.

    (ix) The number “M” of leading zeros is calculated by $M = | \log_2(CodeNum + 2^k)| - k$

    (x) M number of “0” bits are recorded

    (xi) One “1” bit is recorded.

    (xii) INFO is calculated by $INFO = CodeNum + 2^k - 2^{M+k}$

    (xiii) INFO is recorded in the (M+k) bits.

12.2.3.9 ValueWithUnits

12.2.3.9.1 Syntax

```
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 '%'
}
```

12.2.3.9.2 Semantics

This class decodes a value with unit.

12.2.3.10 AnimatedValues

12.2.3.10.1 Syntax

```
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;
      }
    }
  }
}
```

12.2.3.10.2 Semantics

This class decodes a list of animated values.

12.2.3.11 AnimatedValue

12.2.3.11.1 Syntax

```
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;
      }
  }
}
```

12.2.3.11.2 Semantics

This class decodes one animated value.

12.2.3.12 AlignedString

12.2.3.12.1 Syntax

```
class attr_custom_byteAlignedString {
    aligned vluimsbf8 slen;
    byte[slen] UTF-8string;
}
```

12.2.3.12.2 Semantics

This class decodes a string.

12.2.3.13 Fixed16_8

12.2.3.13.1 Syntax

```
class attr_custom_fixed_16_8 {
    int(24) float; // float represented as fixed point with 16 bits mantissa
}
```

12.2.3.13.2 Semantics

This class decodes a fixed point number.

12.2.3.14 Coordinate

12.2.3.14.1 Syntax

```
class attr_custom_coordinate {
  uint(coordBits) coord;
}
```

12.2.3.14.2 Semantics

The `coordBits` value shall be initialised using the initialisation parameters as defined in subclause 6.6.2.2.

12.2.3.15 UpdateValue

12.2.3.15.1 Syntax

```
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
```