International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.763.2
(03/2017)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

IPTV multimedia services and applications for IPTV –
IPTV multimedia application frameworks

Scalable vector graphics for IPTV services

Recommendation ITU-T H.763.2

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Telepresence	H.420–H.429
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Ubiquitous sensor network applications and Internet of Things	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	**H.760–H.769**
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SERVICES AND APPLICATIONS	
Personal health systems	H.810–H.819
Interoperability compliance testing of personal health systems (HRN, PAN, LAN, TAN and WAN)	H.820–H.859
Multimedia e-health data exchange services	H.860–H.869

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

Recommendation ITU-T H.763.2

Scalable vector graphics for IPTV services

Summary

Recommendation ITU-T H.763.2 describes the functionalities of SVG Tiny, which is issued by W3C organization, as one of the multimedia application frameworks for IPTV services. According to different capabilities of terminal devices on supported services and processing performance, SVG Tiny is classified into "Basic Profile" and "Advanced Profile" for different IPTV terminal device models in this Recommendation. Scalable vector graphics (SVG) basic profile is designed for the Internet protocol television (IPTV) terminal device (TD) basic model ITU-T H.721 and mobile model ITU-T H.723, and SVG advanced profile is designed for the IPTV TD full-fledged model ITU-T H.722. SVG advanced profile supports more capabilities of two-dimensional graphics' rendering than SVG basic profile, particularly in styling, animation, multimedia and interactivity.

This Recommendation describes the required aspects of SVG modules, elements, attributes and properties to be supported by those two profiles. It also gives some typical example codes or simulation results.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T H.763.2	2017-03-01	16	11.1002/1000/13210

Keywords

IPTV, multimedia application framework, SVG, XML.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Recommendation ITU-T H.763.2

Scalable vector graphics for IPTV services

1 Scope

This Recommendation describes the functionalities of the scalable vector graphics (SVG) for IPTV services. SVG, as one of those standard multimedia application frameworks described in [ITU-T H.760], provides interoperable use of IPTV services. IPTV terminal devices supporting SVG can provide functionalities of a variety of content and its interactivity, such as audio, video, graphics and text. Expected services include additional data such as text, animation to enrich TV programmes, and two-way portal pages.

This Recommendation describes the profiles, which is a subset of W3C SVG [W3C SVG Tiny], classified into 2 levels to be implemented by IPTV TD Basic model [ITU-T H.721], Mobile model [ITU-T H.723] and Full-fledged model [ITU-T H.722] with different service capabilities. It describes the use of IP-based protocols for transport of SVG and IPTV-related services.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T F.902]	Recommendation ITU-T F.902 (1995), *Interactive services design guidelines.*
[ITU-T H.720]	Recommendation ITU-T H.720 (2008), *Overview of IPTV terminal devices and end systems*.
[ITU-T H.721]	Recommendation ITU-T H.721 (2009), *IPTV terminal devices: Basic model*.
[ITU-T H.722]	Recommendation ITU-T H.722 (2014), *IPTV terminal device: Full-fledged model*.
[ITU-T H.723]	Recommendation ITU-T H.723(2016), *IPTV terminal device: Mobile model.*
[ITU-T H.760]	Recommendation ITU-T H.760 (2009), *Overview of multimedia application frameworks for IPTV services.*
[ITU-T H.762]	Recommendation ITU-T H.762 (2009), *Lightweight interactive multimedia environment (LIME) for IPTV services.*
[ITU-T H.764]	Recommendation ITU-T H.764 (2012), *IPTV services enhanced script language*.
[ITU-T Y.1901]	Recommendation ITU-T Y.1901 (2009), *Requirements for the support of IPTV services.*
[ISO/IEC 16262]	ISO/IEC 16262:2011, *Information technology – Programming languages, their environments and system software interfaces – ECMAScript language specification.*
[W3C SVG Tiny]	W3C Recommendation (2008), *Scalable Vector Graphics (SVG) Tiny 1.2 Specification.*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 Internet protocol television (IPTV) [ITU-T Y.1901]: Multimedia services such as television/video/audio/text/graphics/data delivered over IP-based networks that are managed to support the required level of QoS/QoE, security, interactivity and reliability.

3.1.2 electronic programme guide (EPG) [ITU-T Y.1901]: A structured set of data, intended to provide information on available content that may be accessed by end users.

3.1.3 linear TV [ITU-T Y.1901]: A television service in which a continuous stream flows in real time from the service provider to the terminal device and where the user cannot control the temporal order in which contents are viewed.

3.1.4 video-on-demand (VoD) [ITU-T Y.1901]: A service in which the end user can, on demand, select and view video content and where the end user can control the temporal order in which the video content is viewed (e.g., the ability to start the viewing, pause, fast forward, rewind, etc.).

3.1.5 user agent [ITU-T H.764]: One type of web agent; a piece of software acting on behalf of a person.

3.1.6 terminal device (TD) [ITU-T Y.1901]: An end-user device which typically presents and/or processes the content, such as a personal computer, a computer peripheral, a mobile device, a TV set, a monitor, a VoIP terminal or an audiovisual media player.

3.1.7 user interface (UI) [ITU-T F.902]: Software and hardware components through which a user can interact with a system.

3.1.8 ECMAScript [ISO/IEC 16262]: The programming language defined by ISO/IEC 16262:2011.

3.1.9 LIME-Script [ITU-T H.762]: The subset of ECMAScript defined in this Recommendation that composes lightweight interactive multimedia environment (LIME).

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms.

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IP	Internet Protocol
IPTV SESL	IPTV Service Enhanced Scripting Language
IRI	Internationalized Resource Identifier
LIME	Lightweight Interactive Multimedia Environment for IPTV
MIME	Multipurpose Internet Mail Extensions

SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
SVG DOM	SVG Document Object Model
TD	Terminal Device
uDOM	SVG Micro Document Object Model
UI	User Interface
VoD	Video on Demand
XML	extensible Mark-up Language
XSL	extensible Stylesheet Language
XSLT	XSL Transformations

5 Conventions

In this Recommendation:

– The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this Recommendation is to be claimed.

– The keywords "is prohibited from" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this Recommendation is to be claimed.

– The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

– The keywords "is not recommended" indicate a requirement which is not recommended but which is not specifically prohibited. Thus, conformance with this specification can still be claimed even if this requirement is present.

– The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

6 Introduction

Scalable vector graphic (SVG) [b-W3C SVG] is a language for describing two-dimensional graphics and graphical applications in extensible mark-up language (XML) [b-W3C XML 1.0]. SVG [b-W3C SVG] allows for several types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images, video, audio and text. Graphic objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects.

SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively or via scripting. Sophisticated applications of SVG are possible by use of a supplemental scripting language providing complete access to all elements, attributes and properties to implement multimedia services. This makes SVG one of the Internet protocol television (IPTV) multimedia application framework engines.

SVG Tiny [W3C SVG Tiny] is targeted to resource-limited devices such as mobile phones. Since IPTV terminal device (TD) is a type of resource-restrained terminal device, it may use SVG Tiny as a multimedia application framework engine to implement IPTV services.

According to [ITU-T H.720], IPTV TDs with different capabilities are defined as basic model [ITU-T H.721], mobile model [ITU-T H.723] and full-fledged model [ITU-T H.722]. The basic model and mobile model are only capable of supporting basic IPTV services, such as linear TV, video on demand (VoD) and interactivity, and the full-fledged model may also support supplementary IPTV services such as Internet, medical, communication, etc. As a result of different IPTV services requirements and terminal capabilities, different types of IPTV TDs, which choose SVG Tiny as a multimedia application framework engine, may not support the full set.

In this Recommendation, SVG Tiny [W3C SVG Tiny] is classified into two profiles for different IPTV TDs:

– The profile hereinafter called "Basic Profile" is designed for resource restrained IPTV TD, such as the basic model defined in [ITU-T H.721] and the mobile model in [ITU-T H.723]. The attributes and properties defined in clauses 7 and 9.2 of the basic profile are mandatory for an SVG implementation for these terminal device models.
– The profile hereinafter called "Advanced Profile" is designed for enhanced IPTV TD, such as the full-fledged model [ITU-T H.722]. This profile defines the functional extensions required for supplementary IPTV services. The attributes and properties defined in clauses 7, 8 and 9.2 of the advanced profile are mandatory for an SVG implementation for the full-fledged terminal device model.

7 Basic profile

Since IPTV TD basic model [ITU-T H.721] or mobile model [ITU-T H.723] is only capable of supporting basic audio-visual IPTV services, such as linear TV and VoD, and service navigation with EPG, the basic requirement of SVG's modules, elements, attributes and properties, which is a subset of [W3C SVG Tiny], is defined to be supported in this section as "Basic Profile". The IPTV TD basic model [ITU-T H.721] and mobile model [ITU-T H.723], which chooses SVG as multimedia application framework engine, is recommended to support SVG Basic Profile.

7.1 Document structure

Document object model (DOM) is a language-independent standard object model for representing XML and related formats. Since IPTV TDs are resource-restrained equipment, a subset of DOM Level 2 Specification [b-W3C DOM2], which is called uDOM, is required to be supported.

The goal of the SVG micro document object model (uDOM) definition is to provide an application programming interface (API) that allows access to initial and computed attribute and property values, to reduce the number of interfaces compared to the traditional SVG DOM, and to reduce run-time memory footprint using the necessary features of the core XML DOM.

7.1.1 The common attributes

7.1.1.1 Attributes common to all elements

The following attributes are available on all elements. Some of these elements, such as 'id', 'xml:id', and 'xml:base' may have a direct effect on the structure and rendering of SVG, while others may only affect SVG indirectly, or may be used only for auxiliary processing of SVG content.

– id – This attribute specifies a unique identifier for the element. Because of wider use and compatibility with the legacy content and the existing user agents (including authoring tools), it is recommended that the content intended for use in a MAFR presentation engine should use 'id' instead of 'xml:id'.
– xml:id – This attribute specifies a unique identifier for the element.
– xml:base – This attribute specifies a base internationalized resource identifier (IRI) [b-IETF RFC 3987] other than the base IRI of the document or external entity.

– class – It assigns one or more class names to an element. The element belongs to these classes. A class name may be shared by several element instances.

– role – It assigns one or more role values to an element. The element has these roles. A role value may be shared by several element instances.

– rel – It assigns one or more relationship values to an element. The value of the 'rel' attribute expresses the relationships between two resources.

– rev – It assigns one or more relationship values to an element. The value of the 'rev' attribute expresses the reverse relationships between two resources.

– about – it assigns one or more relationship values to an element. The value of the 'about' attribute is intended to be used for stating the subject of the element's data.

– content – It provides a plain text value that may be suitable for humans, or may be machine-readable, or both, depending on the context. In general, this should only be used to supplement textual child content, or to be used on elements which do not normally take text as child content.

– datatype – It specifies a semantic data type for the content of the element, or the value of the 'content' attribute if one is provided for the element.

– property – it is used for expressing relationships between the element's subject (e.g., the text content of a child node or of an attribute value) and a set of known or referenced properties.

– resource – it associates a resource, typically expressed with an IRI reference, to the element, in a manner that does not normally resolve the IRI reference.

– typeof – it associates one or more data types with the element. This attribute should not be confused with the 'type' attribute, and has no direct effect on the rendering or execution of the element.

7.1.1.2 Attributes for character-content elements

Elements that might contain character data content have attributes 'xml:lang' and 'xml:space' to specify the natural language and whitespace processing of the content.

– xml:lang – This is a standard XML attribute used to specify the language (e.g., English) used in the child text content and attribute values of the element it occurs on.

– xml:space – This is a standard XML attribute used to specify whether white space is preserved in character data. The only possible values are "default" and "preserve".

7.1.2 The 'svg' element

An SVG document fragment consists of any number of SVG elements contained within a 'svg' element, including the 'svg' element.

An SVG document fragment can range from an empty fragment (i.e., no content inside of the 'svg' element), to a very simple SVG document fragment containing a single SVG graphics element such as a 'rect', to a complex, deeply nested collection.

An SVG document fragment can stand by itself as a self-contained file or resource, in which case the SVG document fragment is a SVG document or it can be embedded inline as a fragment within a parent XML document.

An SVG document fragment can only contain one single 'svg' element. This means that 'svg' elements cannot appear in the middle of SVG content. The attribute of 'svg' element for IPTV service is presented in Table 1.

Table 1 – Attributes of the 'svg' element

Attribute	DataType	Default value	Description
version	<string>	1.2	Indicates the SVG language version to which this document fragment conforms. For IPTV TDs, only "1.2" is supported.
baseProfile	<string>	tiny	Describes the minimum SVG language profile that the author believes is necessary to correctly render the content. For IPTV TDs, only "tiny" is supported.
width	<length>	100%	The intrinsic width of the SVG document fragment. A negative value is unsupported. A value of zero disables rendering of the element.
height	<length>	100%	The intrinsic height of the SVG document fragment. A negative value is unsupported. A value of zero disables rendering of the element.
focusable	<string>	auto	Defines whether the grouped elements are focusable. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
contentScriptType	<content-type>	application/ ecmascript	Identifies the default scripting language for the given document.
NOTE – The high performance required attribute will be described in Advanced Profile in this Recommendation.			

The following example shows the simplest SVG content fragment with the 'svg' element.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     width="100%" height="50%">
</svg>
```

7.1.3 The 'g' element

The 'g' element is a container element for grouping those related graphics elements. Grouping constructs, when used in conjunction with the 'desc' and 'title' elements, provide information about document structure and semantics. The attribute of 'g' element for IPTV service is presented in Table 2.

A typical example of the "g" element is shown in Figure 1.

Table 2 – Attributes of the 'g' element

Attribute	DataType	Default value	Description
focusable	<string>	auto	Defines whether the grouped elements are focusable. The possible values are "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
transform	<transform-list> / "none"	none	Specifies a coordinate system transformation to apply to the element it appears on. The possible values are "<transform-list>" | "none". It supports the following values of <transform-list> in Basic Profile: translate(<tx> [<ty>]): specifies a translation by tx and ty. scale(<sx> [<sy>]: specifies a scale operation by sx and sy. For detailed information, please refer to clause 7.6 of [W3C SVG Tiny].
NOTE – The high performance required attribute will be described in Advanced Profile in this Recommendation.			

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" " version="1.2"
     baseProfile="tiny" width="400px" height="120px">
  <g fill="none" stroke="black" stroke-width="3">
    <!-- Draw the axes of the original coordinate system -->
    <line x1="0" y1="1.5" x2="400" y2="1.5"/>
    <line x1="1.5" y1="0" x2="1.5" y2="120"/>
  </g>
  <text x="20" y="20" font-size="18" font-family="Verdana">
        Original coord system
  </text>

  <g transform="translate(50,50)">
    <!-- Establish a new coordinate system from the initial coordinate system
         by 50 user units along x and y axis -->
    <g fill="none" stroke="red" stroke-width="3">
      <line x1="0" y1="0" x2="70" y2="0"/>
      <line x1="0" y1="0" x2="0" y2="70"/>
    </g>
    <text x="20" y="20" font-size="16" font-family="Verdana" stroke="red">
          Translated coord system
    </text>
  </g>
</svg>
```

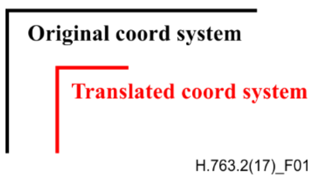


Figure 1 – Example for the "g" element

7.1.4 The 'defs' element

The 'defs' element is a container element for referenced elements. The content model for 'defs' is the same as for the 'g' element; thus, any element that can be a child of a 'g' can also be a child of a 'defs', and vice versa.

There are no attributes defined for the 'defs' element.

The following example shows the simplest SVG content fragment with the 'defs' element.

```
 <?xml version="1.0"?>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
      width="100%" height="100%">
   <defs>
     <linearGradient xml:id="Gradient01">
       <stop offset="0.2" stop-color="#39F"/>
     </linearGradient>
   </defs>
</svg>
```

7.1.5 The 'title' and 'desc' element

Each container element or graphics element in an SVG document may contain one or more of each of the 'title' and 'desc' descriptive elements, which together comprise a sort of heading and summary of the containing element:

– The 'title' element must contain a brief plain text passage representing the title for the container or graphics element containing it.

– The 'desc' element must contain a longer, more detailed plain text description for the container or graphics element containing it.

There are no attributes defined for the 'title' and 'desc' element.

The following example shows the simplest SVG content fragment with the 'title' and 'desc' element.

```
 <?xml version="1.0"?>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
      width="100%" height="100%">
   <title>This is a SVG Profile for IPTV service</title>
   <desc>An example of how the contents of the 'title' and 'desc' elements may be
 presented in a user agent.</desc>
</svg>
```

7.1.6 The 'use' element

Any 'g' or graphics element is potentially a template object that can be reused in the SVG document via a 'use' element. The 'use' element references another element and indicates that the graphical content of that element is to be included and drawn at that given point in the document. The attribute of 'use' element for IPTV service is shown in Table 3.

A typical example of the "use" element is shown in Figure 2.

Table 3 – Attributes of the 'use' element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The x-axis coordinate of one corner of the rectangular region into which the referenced element is placed.
y	<coordinate>	0	The y-axis coordinate of one corner of the rectangular region into which the referenced element is placed.
xlink:href	<IRI>	empty string	An IRI reference to an element/fragment within an SVG document.
focusable	<string>	auto	Defines whether the grouped elements are focusable. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
transform	<transform-list> / "none"	none	See the definition of transform in clause 7.1.3.
NOTE – The high performance required attribute will be described in Advanced Profile in this Recommendation.			

```
 <?xml version="1.0"?>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
      width="10cm" height="3cm">

   <defs>
     <rect xml:id="IPTVRec" width="60" height="10" fill="black"
           fill=opacity="0.5"/>
   </defs>
   <rect x=".1" y=".1" width="99.8" height="29.8"
         fill="none" stroke="blue" stroke-width=".2"/>
   <use x="20" y="10" xlink:href="#IPTVRec" />
</svg>
```

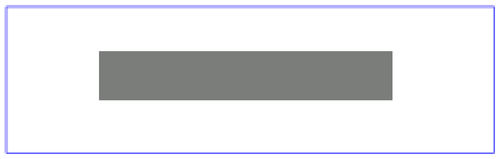


Figure 2 – Example for the "use" element

7.1.7 The 'image' element

The 'image' element indicates that the contents of an image are to be rendered into a given rectangle within the current user coordinate system. In SVG Tiny 1.2, the 'image' must reference content that is a raster image format, such as PNG or JPEG. For detailed information on image format supported by IPTV TDs, please refer to [ITU-T H.721], [ITU-T H.722] and [ITU-T H.723].

The attribute of 'image' element for IPTV service is shown in Table 4.

Table 4 – Attributes of the 'image' element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The x-axis coordinate of one corner of the rectangular region into which the referenced element is placed.
y	<coordinate>	0	The y-axis coordinate of one corner of the rectangular region into which the referenced element is placed.
width	<length>	0	The width of the rectangular region. A negative value is unsupported. A value of zero disables rendering of the element.
height	<length>	0	The height of the rectangular region. A negative value is unsupported. A value of zero disables rendering of the element.
xlink:href	<IRI>	empty string	An IRI reference to an image.
focusable	<string>	auto	Defines whether the element is focusable. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
type	<content-type>	/	A hint about the expected Internet Media Type of the raster image.
NOTE – The high performance required attribute will be described in Advanced Profile in this Recommendation.			

The following example shows the simplest SVG content fragment with the 'image' element.

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     width="10cm" height="10cm">
  <g xml:base="http://a.example.org/aaa/">
   <image xml:id="foo" xlink:href="foo.jpg" width="100" height="100"/>
  </g>
</svg>
```

7.2 Fonts

An SVG font is a font defined using SVG's 'font' element. The purpose of SVG fonts is to allow for delivery of glyph outlines in display-only environments. SVG fonts that accompany web pages must be supported only in browsing and viewing situations.

7.2.1 The 'font' element

The 'font' element defines a character's displaying model on the screen of IPTV TD.

The attributes of this element are presented in Table 5.

Table 5 – Attributes of the "font" element

Attribute	DataType	Default value	Description
horiz-origin-x	<number>	0	The X-coordinate in the font coordinate system of the origin of a glyph to be used when drawing horizontally oriented text.
horiz-adv-x	<number>	0	The default horizontal advance after rendering a glyph in horizontal orientation. Glyph widths are required to be non-negative, even if the glyph is typically rendered right-to-left, as in Hebrew and Arabic scripts.
NOTE – Each 'font' element must have a 'font-face' child element which describes various characteristics of the font.			

The following is a typical example of the "font" element.

```
 <font xml:id="Font1" horiz-adv-x="1000">
       <font-face font-family="Super Sans" font-weight="bold" font-style="normal">
       </font-face>
</font>
```

7.2.2 The 'glyph' element

The 'glyph' element defines the graphics for a given glyph. The coordinate system for the glyph is defined by the various attributes in the 'font' element.

The graphics that make up the 'glyph' consist of a single path data specification within the 'd' attribute.

Table 6 – Attributes of the "glyph" element

Attribute	DataType	Default value	Description
d	string	/	The definition of the outline of a glyph, using the same syntax as for the 'd' attribute on a 'path' element.

7.2.3 The 'missing-glyph' element

The 'missing-glyph' element defines a graphic to use if there is an attempt to draw a glyph from a given font and the given glyph has not been defined. The attributes on the 'missing-glyph' element have the same meaning as the corresponding attributes on the 'glyph' element.

7.2.4 The 'font-face' element

The 'font-face' element is an XML structure which is directly corresponded to the 'font-face' facility in CSS2. It can be used to describe the characteristics of any font, SVG font or otherwise.

Table 7 – Attributes of the "font-face" element

Attribute	DataType	Default value	Description
font-family	<string>	system's default font	A prioritized list of font family names or generic family names. Possible values of generic families are: "'serif" | "sans-serif" | "cursive"' | "fantasy" | "'monospace".
font-style	<string>	normal	Selects the style of font within a font family. Possible values are: "normal" | "italic" | "oblique".
font-weight	<string>	normal	Selects the weight of the font. Possible values are: "normal" | "bold" | "bolder" | "lighter" | "100" | "200" | "300" | "400" | "500" | "600" | "700" | "800" | "900".

The following is a typical example of the "font-face" element.

```
<font-face font-family="Super Sans" font-weight="bold" font-style="normal"">
</font-face>
```

7.3 Shape

Shape is a graphics element that comprises a defined combination of straight lines and curves on the screen of IPTV TD.

Shape can use the styling properties of Painting inside its elements, see clause 9.2. It can also use the common attribute 'transform', 'focusable' and Navigation. See the definition of transform in clause 7.1.3 and the definition of Navigation in clause 7.6.3.

7.3.1 The 'rect' element

The element 'rect' defines a rectangle which is axis-aligned with the current user coordinate system. For IPTV service, the "coordinate system" indicates the video output resolution of the IPTV TD.

The attributes of this element are defined in Table 8.

Table 8 – Attributes of the "rect" element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The left-top point's x-axis coordinate of the rectangle, e.g., x="100".
y	<coordinate>	0	The left-top point's y-axis coordinate of the rectangle, e.g., y="100".
width	<length>	0	The width of the rectangle, e.g., width="200". The value "0" indicates that the rendering of the element is disabled.
height	<length>	0	The height of the rectangle, e.g., width="50". The value "0" indicates that the rendering of the element is disabled.
rx	<length>	0	The x-axis radius of the ellipse used to round off the corners of the rectangle, e.g., rx="10". The value "0" indicates that no rounding corner is specified.

Table 8 – Attributes of the "rect" element

Attribute	DataType	Default value	Description
ry	<length>	0	The y-axis radius of the ellipse used to round off the corners of the rectangle, e.g., ry="10". The value "0" indicates that no rounding corner is specified.
NOTE 1 – If the "rx" and "ry" are not provided, it indicates no rounding corner is specified. NOTE 2 – If the "rx" is provided but not for "ry", the same value of "rx" should be processed by the user agent for "ry". NOTE 3 – If the "ry" is provided but not for "rx", the same value of "ry" should be processed by the user agent for "rx".			

The following is a typical example of the "rect" element.

```
<rect x="100" y="100" width="200" height="100" rx="10" ry="10" />
```

7.3.2 The 'circle' element

The 'circle' element defines a circle based on a centre point and a radius.

The attributes of the 'circle' element are given in Table 9.

Table 9 – Attributes of the "circle" element

Attribute	DataType	Default value	Description
cx	<coordinate>	0	The x-axis coordinate of the centre of the circle, e.g., cx="100".
cy	<coordinate>	0	The y-axis coordinate of the centre of the circle, e.g., cy="100".
r	<length>	0	The radius of the circle, e.g., r="50". The value "0" indicates the rendering of the element is disabled.

The following is a typical example of the "circle" element.

```
<circle cx="100" cy="100" r="50" />
```

7.3.3 The 'line' element

The 'line' element defines a line segment that starts at one point and ends at another.

The attributes of this element are given in Table 10.

Table 10 – Attributes of the "line" element

Attribute	DataType	Default value	Description
x1	<coordinate>	0	The x-axis coordinate of the start point, e.g., x1="100".
y1	<coordinate>	0	The y-axis coordinate of the start point, e.g., y1="100".
x2	<coordinate>	0	The x-axis coordinate of the end point, e.g., x2="200".
y2	<coordinate>	0	The y-axis coordinate of the end point, e.g., y1="200".
NOTE – If the "x2" and "y2" is the same as "x1" and "y1", it indicates that the rendering of the element is disabled.			

The following is a typical example of the "line" element.

```
<line x1="100" y1="100" x2="200" y2="200" />
```

7.3.4 The 'polyline' element

The "polyline" element defines a set of connected straight line segments. Typically, "polyline" elements define the open shapes.

The attributes of this element are given in Table 11.

A typical example of the "polyline" element is shown in Figure 3.

Table 11 – Attributes of the "polyline" element

Attribute	DataType	Default value	Description
points	<string>	empty string	The list of points that make up the polyline. Each point in the list is described in a pair of x-axis and y-axis coordinate separated by a comma ",". Different points are separated by blank " ". E.g., points="10,10 100,100 150,200 200,200". The empty string "" indicates the rendering of the element is disabled.
NOTE – If an odd number of coordinates is provided, then the element is treated as if the attribute had not been specified.			

```
<polyline points="30,35 45,55 75,55 90,35 75,40 60,20 45,40 " fill="#FFCC00"
stroke="#666666"/>
```

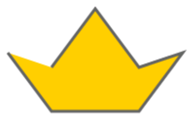


Figure 3 – Example for the "polyline" element

7.3.5 The "polygon" element

The "polygon" element defines a closed shape consisting of a set of connected straight line segments.

The attributes of this element are given in Table 12.

A typical example of the "polygon" element is shown in Figure 4.

Table 12 – Attributes of the "polygon" element

Attribute	DataType	Default value	Description
points	<string>	empty string	The list of points that make up the polygon. Each point in the list is described in a coordinate-pair of x-axis and y-axis separated by a comma ",". Different points are separated by blank " ". E.g., points="10,10 100,200 150,100 100,150". The empty string "" indicates the rendering of the element is disabled.
NOTE – If an odd number of coordinates is provided, then the element is treated as if the attribute had not been specified.			

```
<polygon  points="30,35  45,55  75,55  90,35  75,40  60,20  45,40  "  fill="#FFCC00"
        stroke="#666666"/>
```

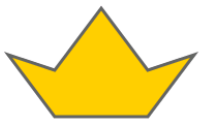


Figure 4 – Example for the "polygon" element

7.4 Text

Text that is to be rendered as part of an SVG document fragment is specified for using text content elements, inside which the characters to be drawn are expressed as XML character data. SVG's text content elements are rendered like other graphics elements. Therefore, the same coordinate system and rendering features can be applied to text elements.

Text elements can apply the styling properties of Fonts (i.e., 'font-family', 'font-style', 'font-weight', 'font-size') and properties of Text (i.e., 'text-anchor', 'display-align', 'line-increment') to text content rendering. These properties are used as attributes in Text elements. See clause 9.2.

7.4.1 The 'text' element

The 'text' element defines a graphics element consisting of text. Each 'text' element causes a single string of text to be rendered and performs no automatic line breaking or word wrapping.

The attributes of this element are given in Table 13.

Table 13 – Attributes of the "text" element

Attribute	DataType	Default value	Description
x	<list-of-coordinates>	0	The corresponding list of absolute X coordinates for the glyphs corresponding to the characters within this element. For IPTV TD basic model [ITU-T H.721], the recommended minimum value range is 0-720. For full-fledged model [ITU-T H.722], the recommended rang is 0-1920.

Table 13 – Attributes of the "text" element

Attribute	DataType	Default value	Description
y	<list-of-coordinates>	0	The corresponding list of absolute Y coordinates for the glyphs corresponding to the characters within this element. For IPTV TD basic model [ITU-T H.721], the recommended minimum value range is 0-576. For full-fledged model [ITU-T H.722], the recommended rang is 0-1080.
focusable	<string>	auto	Defines whether the element is focusable. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
editable	<string>	none	Indicates whether the text can be edited. Possible values are: none: editing facilities are not provided for the content. simple: user agent must provide a way to edit the content which is not hidden or removed from the rendering tree.
NOTE – If a single <coordinate> is provided, then the value represents for the current text position for rendering the glyphs that correspond to the first character within this element or any of its descendants.			

The following is a typical example of 'text' element.

```
<text x="100" y="100" font-family="sans-serif" font-weight="bold" font-
     style="normal" font-size="55">
     Text using embedded font
         </text>
```

7.4.2 The 'tbreak' element

The 'tbreak' element is an empty element that forcibly breaks the current line of text, even if the current line of text is empty (i.e., multiple consecutive 'tbreak' elements each cause a line break). It has no attributes aside from the standard core and conditional attributes.

7.5 Painting

Painting is applied in successive operations to graphics elements including shapes, text, raster image and video. SVG Tiny supports two built-in types of fill and stroke rendering operations, i.e., solid colour and gradients.

7.5.1 The 'solidColor' element

The 'solidColor' element is a paint server that provides a single colour with opacity. It can only be referenced using the 'fill' and 'stroke' properties.

This element applies 'solid-color' property and 'solid-opacity' property to define the colour and opacity of painting. See clause 9.2.

A typical example of 'solidColor' element is shown in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
     version="1.2" baseProfile="tiny" width="480" height="360">
   <title>'solidColor' example</title>
   <defs>
      <solidColor xml:id="solidMaroon" solid-color="maroon" solid-opacity="0.7"/>
   </defs>
   <g>
     <rect fill="none" stroke="blue" x="1" y="1" width="478" height="358"/>
     <circle fill="url(#solidMaroon)" cx="100" cy="100" r="60"/>
     <rect fill="url(#solidMaroon)" x="300" y="180" width="120" height="120"/>
   </g>
</svg
```

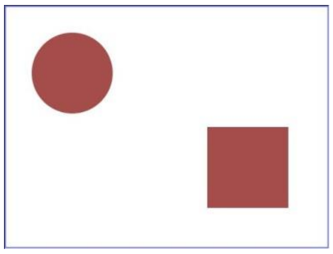


Figure 5 – Example for the "solidColor" element

7.6 Interactivity

SVG content can be interactive (i.e., responsive to user-initiated events) by utilizing the following features in the SVG language:

– User-initiated actions such as a key-press can cause timed elements (such as Animation elements in Basic Profile) to start or stop, scripts to execute or 'listener' elements to trigger 'handler' elements.

– The user can initiate hyperlinks to new web pages (see the 'a' element) by actions such as a stylus click on a particular graphics element.

7.6.1 Supported events in IPTV

The following aspects of SVG are affected by events:

– The SVG uDOM enables a script to register event listeners so that the script can be invoked when a given event occurs.

– The ev:event attribute on the handler element specifies for which event the handler should trigger.

– Timed elements can be defined to begin or end based on events.

Table 14 – Supported events in IPTV service

Event identifier {event-namespace, eventlocalname}	Description	DOM3 event category	Animation event name	uDOM interface
{"http://www.w3.org/2001/xmlevents", "mouseout"}	Occurs when the pointing device is moved away from an element.	Mouse Event	mouseout	Mouse Event

Table 14 – Supported events in IPTV service

Event identifier {event-namespace, eventlocalname}	Description	DOM3 event category	Animation event name	uDOM interface
{"http://www.w3.org/2001/ xmlevents", "DOMFocusIn"} SVG 1.2 alias: {"http://www.w3.org/2001/ xml-events", "focusin"} (see Notes below)	Occurs when an element receives focus.	UIEvent	focusin	UIEvent
{"http://www.w3.org/2001/ xmlevents", "DOMFocusOut"} SVG 1.2 alias: {"http://www.w3.org/2001/ xml-events", "focusout"} (see Notes below)	Occurs when an element loses focus.	UIEvent	focusout	UIEvent
{"http://www.w3.org/2001/ xmlevents", "DOMActivate"} SVG 1.2 alias: {"http://www.w3.org/2001/ xml-events", "activate"} (see Notes below)	Occurs when an element is activated.	UIEvent	activate	UIEvent
{"http://www.w3.org/2001/ xmlevents", "click"}	Occurs when the pointing device button is clicked over an element. A click is defined as a mousedown and mouseup over the same screen location. The sequence of these events is: mousedown, mouseup, click. If multiple clicks occur at the same screen location, the sequence repeats with the detail attribute incrementing with each repetition.	Mouse Event	click	Mouse Event
{"http://www.w3.org/2001/ xmlevents", "mousedown"}	Occurs when the pointing device button is pressed over an element.	Mouse Event	mousedown	Mouse Event
{"http://www.w3.org/2001/ xmlevents", "mouseup"}	Occurs when the pointing device button is released over an element.	Mouse Event	mouseup	Mouse Event
{"http://www.w3.org/2001/ xmlevents", "mouseover"}	Occurs when the pointing device is moved onto an element.	Mouse Event	mouseover	Mouse Event
{"http://www.w3.org/2001/ xmlevents", "mousemove"}	Occurs when the pointing device is moved while it is over an element.	Mouse Event	mousemove	Mouse Event
{"http://www.w3.org/2001/ xmlevents", "mouseout"}	Occurs when the pointing device is moved away from an element.	Mouse Event	mouseout	Mouse Event

Table 14 – Supported events in IPTV service

Event identifier {event-namespace, eventlocalname}	Description	DOM3 event category	Animation event name	uDOM interface
{"http://www.w3.org/2001/ xmlevents", "textInput"}	One or more characters have been entered.	TextEvent	none	TextEvent
{"http://www.w3.org/2001/ xmlevents", "keydown"}	A key is pressed down. (The normative definition of this event is the description in the DOM3 Events specification.)	Keyboard Event	none	Keyboard Event
{"http://www.w3.org/2001/ xmlevents", "keyup"}	A key is released. (The normative definition of this event is the description in the DOM3 Events specification.)	Keyboard Event	none	Keyboard Event
{"http://www.w3.org/2001/ xmlevents", "load"} Deprecated backwards compatibility alias: {"http://www.w3.org/ 2001/xml-events", "SVGLoad"} (see Notes below).	The event is triggered at the point at which the user agent has fully parsed the element and its descendants and is ready to act appropriately upon that element, such as being ready to render the element to the target device. Referenced external resources that are required must be loaded, parsed and ready to render before the event is triggered. Optional external resources are not required to be ready for the event to be triggered.	HTML Event	load	Event
{"http://www.w3.org/2001/ xmlevents", "resize"} Deprecated backwards compatibility alias: {"http://www.w3.org/ 2001/xml-events", "SVGResize"} (see Notes below).	Occurs when a document view is being resized. This event is only applicable to 'svg' elements and is dispatched after the resize operation has taken place. The target of the event is the 'svg' element.	HTML Event	resize	Event
{"http://www.w3.org/2001/ xmlevents", "scroll"} Deprecated backwards compatibility alias: {"http://www.w3.org/2001/ xml-events", "SVGScroll"} (see Notes below).	Occurs when a document view is being shifted along the X or Y or both axis, either through a direct user interaction or any change on the 'currentTranslate' property available on SVGSVGElement interface. This event is only applicable to 'svg' elements and is dispatched after the shift modification has taken place. The target of the event is the 'svg' element.	HTML Event	scroll	Event

Table 14 – Supported events in IPTV service

Event identifier {event-namespace, eventlocalname}	Description	DOM3 event category	Animation event name	uDOM interface
{"http://www.w3.org/2001/xmlevents", "zoom"} Deprecated backwards compatibility alias: {"http://www.w3.org/2001/xml-events", "SVGZoom"} (see Notes below).	Occurs when the zoom level of a document view is being changed, either through a direct user interaction or any change to the 'currentScale' property available on SVGSVGElement interface. This event is only applicable to 'svg' elements and is dispatched after the zoom level modification has taken place. The target of the event is the 'svg' element.	DOM3's SVG Events	zoom	Event
{"http://www.w3.org/2001/xmlevents", "beginEvent"}	Occurs when an animation element begins. For details, see the description of the Events and event model in [b-W3C SMIL].	DOM3's Timing Events	beginEvent	TimeEvent
{"http://www.w3.org/2001/xmlevents", "endEvent"}.	Occurs when an animation element ends. For details, see the description of the Events and event model in [b-W3C SMIL].	DOM3's Timing Events	endEvent	TimeEvent
{"http://www.w3.org/2001/xmlevents", "repeatEvent"}	Occurs when an animation element repeats. It is raised each time the element repeats, after the first iteration. For details, see the description of the Events and event model in [b-W3C SMIL].	DOM3's Timing Events	repeat	TimeEvent
{"http://www.w3.org/2001/xmlevents", "wheel"}	Occurs when a rotational input device has been activated	UIEvent	none	Wheel Event
{"http://www.w3.org/2000/svg", "preload"}	A load operation has begun.	none	none	Progress Event
{"http://www.w3.org/2000/svg", "loadProgress"}	Progress has occurred in loading a given resource.	none	none	Progress Event
{"http://www.w3.org/2000/svg", "postload"}	A load operation has completed.	none	none	Progress Event
{"http://www.w3.org/2001/xmlevents", "timer"}	Occurs when the specified timer interval has elapsed for a timer. This event is triggered only by 'enabled' timers in the current global execution context of the SVG document (i.e., for timers which have been instantiated via the SVGGlobal interface and started via the start () method of the SVGTimer interface).	none	none	Event
{"http://www.w3.org/2000/svg", "connection Connected"}	Occurs when a connection has been established. No context information is available.	none	none	Connection Event

Table 14 – Supported events in IPTV service

Event identifier {event-namespace, eventlocalname}	Description	DOM3 event category	Animation event name	uDOM interface
{"http://www.w3.org/2000/svg", "connectionClosed"}	Occurs when a connection has been closed. No context information is available.	none	none	Connection Event
{"http://www.w3.org/2000/svg", "connectionError"}	Occurs when an error happens during the lifetime of a connection. Additional context information is available in the errorCode field.	none	none	Connection Event
{"http://www.w3.org/2000/svg", "connectionDataSent"}	Occurs when data has been successfully transmitted. No context information is available.	none	none	Connection Event
{"http://www.w3.org/2000/svg", "connectionData Received"}	Occurs when data has been received on the connection. Additional context information is available on the receivedData field.	none	none	Connection Event
"remoteControlKeyDown"	A key of the remote control is pressed down.	none	none	none
"remoteControlKeyUp"	A key of the remote control is released.	none	none	none
"somatoSensoryRemote ControlMove"	Occurs when the somato sensory remote control moves.	none	none	none
"somatoSensoryRemote ControlKeyDown"	A key of the somato sensory remote control is pressed down.	none	none	none
"somatoSensoryRemote ControlKeyUp"	A key of the somato sensory remote control is released.	none	none	none

The following should be noted:

1) Unqualified event names, including SVG 1.1 event names, are assumed to be in the "http://www.w3.org/2001/xml-events" namespace. This allows SVG 1.1 content (which did not have a notion of namespaced events) to be upwardly compatible with SVG 1.2 (which adds a notion of namespaced events). Therefore, the SVG 1.1 "SVGZoom" event becomes the {"http://www.w3.org/2001/xml-events", "SVGZoom"} event in SVG 1.2.

2) In order to unify event names with other W3C languages, SVG 1.2 deprecates some of the SVG 1.1 event names. (The term "deprecate" in this case means that user agents which are compatible with both SVG 1.1 and SVG 1.2 must support both the old deprecated event names and the new event names, but an SVG 1.2-only user agent should support only the new event names. Content creators who are making content that targets SVG 1.2 should use the new event names, not the deprecated event names.) The specifics are as follows:
   – "SVGLoad" event is deprecated in favour of {"http://www.w3.org/2001/xml-events", "load"}
   – "SVGResize" event is deprecated in favour of {"http://www.w3.org/2001/xml-events", "resize"}
   – "SVGScroll" event is deprecated in favour of {"http://www.w3.org/2001/xml-events", "scroll"}

– "SVGZoom" event is deprecated in favour of {"http://www.w3.org/2001/xml-events", "zoom"}

In cases where the event name from SVG 1.1 differs from the DOM3 event name, but the SVG event name is more user-friendly (e.g., "focusin" is more user friendly than "DOMFocusIn"), the SVG 1.1 event name is not deprecated but instead retained as an alias for the DOM3 event name. Therefore, the following aliases shall be available:

– {"http://www.w3.org/2001/xml-events", "focusin"} is equivalent to {"http://www.w3.org/2001/xml-events", "DOMFocusIn"}

– {"http://www.w3.org/2001/xml-events", "focusout"} is equivalent to {"http://www.w3.org/2001/xml-events", "DOMFocusOut"}

– {"http://www.w3.org/2001/xml-events", "activate"} is equivalent to {"http://www.w3.org/2001/xml-events", "DOMActivate"}.

3) In almost all cases, the event identifiers for SMIL timing attributes (e.g., the 'begin' and 'end' attributes on animation elements) are required to be the localname for the events, where the assumed namespace for the event is "http://www.w3.org/2001/xml-events". For example, to indicate that an animation should begin with a "click" event, then the begin attribute would be specified as begin="click". Some exceptions to this rule are necessary for SVG 1.1 backwards compatibility and to make it easier to integrate SVG with other W3C languages. The following events shall use the following strings rather than their localname as their event identifier for SMIL timing attributes:

– The DOMFocusIn event uses the string "focusin"

– The DOMFocusOut event uses the string "focusout"

– The DOMActivate event uses the string "activate".

For backwards-compatibility and to make it easier to integrate SVG with other W3C languages, with SVG 1.2 SMIL timing attributes must accept the following aliases for event identifiers:

– "load" and "SVGLoad" are aliases for each other

– "resize" and "SVGResize" are aliases for each other

– "scroll" and "SVGScroll" are aliases for each other

– "zoom" and "SVGZoom" are aliases for each other.

7.6.2 User interface events

Concerning user agents who support interactivity, it is common for authors to define SVG documents such that they are responsive to user interface events. Among the set of possible user events are pointer events, keyboard events, document events and events triggered by remote control.

In response to user interface (UI) events, the author might start an animation, perform a hyperlink to another web page, highlight parts of the document (e.g., change the colour of the graphics elements which are under the pointer), initiate a "roll-over" (e.g., cause some previously hidden graphics elements to appear near the pointer) or launch a script which communicates with a remote database.

7.6.2.1 Pointer events

User interface events that occur because of user actions performed on a pointer device are called pointer events. Devices for IPTV that can trigger pointer events contain mouse and sonata sensory remote control.

IPTV systems support pointer devices such as a mouse or a somato sensory remote control. With a mouse, pointer events consist of actions such as mouse movements and mouse clicks. With a somato sensory remote control, pointer events consist of actions such as waving the remote control or pointing to the screen with the remote control. When pointing to the screen with the remote control, the cursor will move to the position pointed to.

SVG user agent supports the 'pointer-events' property to specify under what circumstances a given graphics element can be the target element for a pointer event. See the definition of this property in clause 9.2.

7.6.2.2 Text events

User interface events that occur because of user actions that generate text are called text events. They are usually generated by a keyboard or a remote control. DOM 3 Text Events only requires that a string of Unicode characters is returned, making it independent of the input device used.

7.6.2.3 Key events

User interface events that occur because of user actions that generate key presses (as opposed to text – for example, function keys, key presses for a game, etc.) are called key events. Key events could be triggered by a keyboard or a remote control, including common remote control and somato sensory remote control.

7.6.3 Navigation attributes

In many cases, such as text editing, the user is required to place focus on a particular element, ensuring that input events, such as keyboard input, are sent to that element. Attribute 'focusable' is used to define if an element can get keyboard focus (i.e., receive keyboard events) and be a target for field-to-field navigation actions.

System-dependent input facilities (e.g., the tab key on most desktop computers) should be supported to allow navigation between elements that can obtain focus (i.e., elements for which the value of the 'focusable' attribute is 'true'). SVG specifies some rules for document focus ring, focus navigation behaviour (including referenced content), etc. For detailed information, please refer to clause 13.13.1 of [W3C SVG Tiny].

The attributes of Navigation are given in Table 15.

A typical example of focus navigation is shown in Figure 6. This example illustrates how to control the focus order between several focusable elements.

Table 15 – Navigation attributes of Interactivity

Attribute	DataType	Default value	Description
nav-next	<FuncIRI> / <string>	auto	Specifies the next element in the focus ring. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-prev	<FuncIRI> / <string>	auto	Specifies the previous element in the focus ring. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-up	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in an upward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-up-right	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in an up-and-rightward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-right	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in a rightward direction. The possible values are: "<FuncIRI>" | "auto" | "self".

Table 15 – Navigation attributes of Interactivity

Attribute	DataType	Default value	Description
nav-down-right	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in a down-and-rightward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-down	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in a downward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-down-left	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in a down-and-leftward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-left	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in a leftward direction. The possible values are: "<FuncIRI>" | "auto" | "self".
nav-up-left	<FuncIRI> / <string>	auto	Specifies an element to receive focus when navigating in an up-and-leftward direction. The possible values are: "<FuncIRI>" | "auto" | "self".

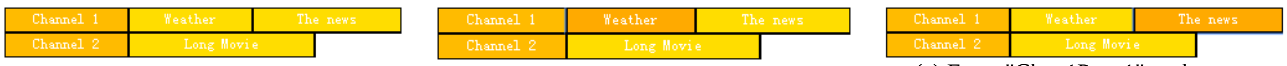
```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
     version="1.2" baseProfile="tiny" width="220px" height="40px">

<!-- List of available channels -->
   <rect x="0" y="0" width="100" height="20" fill="#fb0" stroke="#000"
         stroke-width="2" />
  <text x="50" y="13" font-size="8" font-family="Arial Black"
        fill="#fff" text-anchor="middle">Channel 1</text>
  <rect x="0" y="20" width="100" height="20" fill="#fb0" stroke="#000"
        stroke-width="2" />
  <text x="50" y="33" font-size="8" font-family="Arial Black"
        fill="#fff" text-anchor="middle">Channel 2</text>

<!-- List of programs for channel nb 1 -->
  <g xml:id="Chan1Prog1" focusable="true" nav-left="self"
     nav-right="url(#Chan1Prog2)" nav-up="self" nav-down="url(#Chan2Prog1)">
    <rect x="100" y="0" width="100" height="20" fill="#fd0" stroke="#000"
          stroke-width="2">
      <set attributeName="fill" begin="Chan1Prog1.focusin"
           end="Chan1Prog1.focusout" to="#fa0"/>
    </rect>
    <text x="150" y="13" font-size="8" font-family="Arial Black"
          fill="#fff" text-anchor="middle">Weather</text>
  </g>
  <g xml:id="Chan1Prog2" focusable="true" nav-left="url(#Chan1Prog1)"
     nav-right="self" nav-up="self" nav-down="auto" nav-next="self">
    <rect x="200" y="0" width="120" height="20" fill="#fd0" stroke="#000"
          stroke-width="2">
      <set attributeName="fill" begin="Chan1Prog2.focusin"
           end="Chan1Prog2.focusout" to="#fa0"/>
    </rect>
    <text x="260" y="13" font-size="8" font-family="Arial Black"
          fill="#fff" text-anchor="middle">The news</text>
  </g>

<!-- List of programs for channel nb 2 -->
  <g xml:id="Chan2Prog1" focusable="true" nav-left="self" nav-right="auto"
     nav-up="url(#Chan1Prog1)" nav-down="auto" nav-prev="url(#Chan1Prog1)"
     nav-next="auto">
    <rect x="100" y="20" width="150" height="20" fill="#fd0" stroke="#000"
          stroke-width="2">
      <set attributeName="fill" begin="Chan2Prog1.focusin"
```

```
          end="Chan2Prog1.focusout" to="#fa0"/>
    </rect>
    <text x="175" y="33" font-size="8" font-family="Arial Black"
          fill="#fff" text-anchor="middle">Long Movie</text>
  </g>
</svg>
```



(a) – Initial state

(b) "Chan1Prog1" receives focus

(c) From "Chan1Prog1" to the next focusable "Chan1Prog2"

Figure 6 – Example for focus navigation

7.6.4 Scripting

By utilizing the feature of scripts, SVG content can be interactive in IPTV service. Scripting elements define the execution logic of scripts responding to user-initiated events.

7.6.4.1 The 'script' element

A 'script' element may either contain or point to executable content, which can be either in the form of an embedded script code or in the form of external resource.

The attribute of 'script' element for IPTV service is as shown in Table 16.

A typical example of the "script" element is shown in Figure 7.

Table 16 – Attributes of the 'script' element

Attribute	DataType	Default value	Description
type	<content-type>	application / ecmascript	Identifies the programming language for this element. Implementations in conformance with SVG basic profile for IPTV are mandatory to support scripting language as defined in SESL core script profile [ITU-T H.764].
xlink:href	<IRI>	empty string	An IRI refers to an external resource containing the script code. If it contains an invalid IRI reference, the 'script' element will not execute any script.
NOTE 1 – If this element has both an 'xlink:href' attribute and child character data, the executable content for the script is retrieved from the IRI of the 'xlink:href' attribute, and the child content is not added to the scripting context. NOTE 2 – IPTV service supports LIME-Script and IPTV SESL based on ECMAScript language. For detailed information, please refer to [ITU-T H.762] and [ITU-T H.764].			

```
<?xml version="1.0"?>
<svg width="600px" height="450px" version="1.2" baseProfile="tiny"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:ev="http://www.w3.org/2001/xml-events">

<!-- ECMAScript to change the radius with each click -->
  <script type="application/ecmascript"> <![CDATA[
    function circle_click(evt) {
      var circle = evt.target;
      var currentRadius = circle.getFloatTrait("r");
      if (currentRadius == 100)
        circle.setFloatTrait("r", currentRadius*2);
      else
```

```
          circle.setFloatTrait("r", currentRadius*0.5);
      }
    ]]> </script>

    <rect x="1" y="1" width="598" height="448" fill="none" stroke="blue"/>
    <circle cx="300" cy="225" r="100" fill="gray">
      <handler type="application/ecmascript" ev:event="click">
            circle_click(evt);
       </handler>
    </circle>
    <text x="240" y="240" font-size="58">Start</text>
  </svg>
```

Start
Start

(a) Initial state (b) After the first click

Figure 7 – Example for the "script" element

7.6.4.2 The 'listener' element

The 'listener' element supports XML Events. It is used to declare event listeners and register them to the specific SVG uDOM nodes.

The attribute of 'listener' element is shown in Table 17.

Table 17 – Attributes of the 'listener' element

Attribute	DataType	Default value	Description
event	XML-NMTOKEN	/	Specifies one or more event types for which the listener is being registered. For detailed information of supported events in IPTV service, please refer to clause 7.6.1.
observer	<IDREF>	/	Specifies the id of one or more elements with which the event listener is to be registered.
target	<IDREF>	/	Specifies the id of one or more target elements of the event.
handler	<IRI>	/	Specifies the URI reference of a resource that defines the action that should be performed if the event reaches the observer.
propagate	<string>	Continue	Specifies whether the event is allowed to continue on its path after processing all listeners at the current node. The possible values are: "stop" | "continue".
defaultAction	<string>	Perform	Specifies whether the default action for the event should be performed or not after processing of all the listeners for the event. The possible values are: "cancel" | "perform".

7.6.4.3 The 'handler' element

The 'handler' element is similar to the 'script' element. It contains inline or referenced codes to be executed by user agent.

The attribute of 'handler' element is shown in Table 18

Table 18 – Attributes of the 'handler' element

Attribute	DataType	Default value	Description
type	<content-type>	Application / ecmascript	Identifies the programming language for this element.
xlink:href	<IRI>	Empty string	An IRI reference to an external resource containing the script code. If it contains an invalid IRI reference, the 'script' element will not execute any script.
en:event	<XML-NMTOKEN>	/	Specifies the name of the event to handle. This attribute is in the XML Events namespace.

A typical example of the "listener" and "handler" elements is shown in Figure 8. In this example, the 'listener' element is used to specify the 'observer' and 'handler' for a particular XML Event. And the 'handler' element is reused on several objects for execution in response to the event.

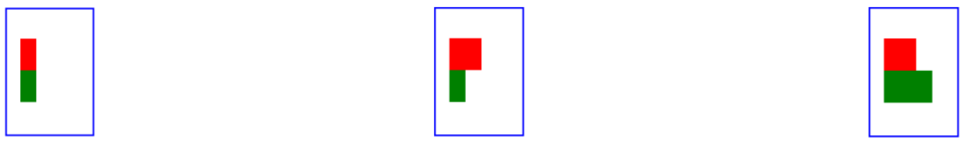
```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     xmlns:ev="http://www.w3.org/2001/xml-events">

  <rect xml:id="theRect1" x="10" y="20" width="10" height="20" fill="red"/>
  <rect xml:id="theRect2" x="10" y="40" width="10" height="20" fill="green"/>
  <rect x="1" y="1" width="55" height="80" fill="none" stroke="blue"/>

  <ev:listener event="click" observer="theRect1" handler="#theClickHandler"/>
  <ev:listener event="click" observer="theRect2" handler="#theClickHandler"/>

  <handler xml:id="theClickHandler" type="application/ecmascript">
    var theRect = evt.target;
    var width = theRect.getFloatTrait("width");
    theRect.setFloatTrait("width", (width+10));
  </handler>

</svg>
```



(a) Initial state (b) After the first click of "theRect1" (c) After the second click of "theRect2"

Figure 8 – Example for the "listener" and "handler" elements

7.7 Animation

Animation elements are used to define motion paths as fade-in or fade-out effects and allow objects to grow, shrink, spin or change color. SVG supports the ability to change vector graphics over time in the following ways:

– Using SVG's animation elements. SVG document fragments can describe time-based modifications to the document's elements.

– Using the SVG uDOM. The SVG uDOM conforms to the document object model Level 3 specification [b-W3C DOM3]. Some of the SVG attributes are accessible to scripting and by manipulating them many kind of animations can be achieved.

7.7.1 The common attributes

Animation supports the common attributes for all animation elements described in Table 19. These attributes can be used to identify the target element or attribute or property for an animation, control the timing of the animation and define animation values over time. The timing attributes can also be applied to media elements (i.e., 'audio' and 'video').

Table 19 – Common attributes of animation elements

Attribute	DataType	Default value	Description
xlink:href	<IRI>	Empty string	An IRI reference to the element which is the target of this animation and will be modified over time. If this attribute is not provided, then the target element will be the immediate parent element of the current animation element.
attributeName	<QName>	/	Specifies the name of the target attribute. A qualified name indicates the XML namespace for the attribute.
attributeType	<string>	Auto	Specifies the namespace in which the target attribute and its associated values are defined. The possible values are: 'CSS' | 'XML' | 'auto'.
begin	<begin-value-list>	An offset value of '0'	Defines when the element should begin, i.e., become active. The attribute value is a semicolon separated list of timing specifier values. For detailed information, please refer to clause 16.2.8 of [W3C SVG Tiny].
dur	<clock-value> / 'media' / 'indefinite'	Indefinite	Specifies the simple duration. The possible values are: '<clock-value>' | 'media' | 'indefinite'.
end	<end-value-list>	An offset value of '0'	Defines an end value for the animation that can constrain the active duration.
min	<clock-value> / 'media'	0	Specifies the minimum value of the active duration. The possible values are: '<clock-value>' | 'media'.
max	<clock-value> / 'media'	/	Specifies the maximum value of the active duration. The possible values are: '<clock-value>' | 'media'.
restart	<string>	Always	Specifies when to restart the animation. The possible values: "always" | "whenNotActive" | "never".
repeatCount	<number> / "indefinite"	1	Specifies the number of iterations of the animation function. The possible values: "<number>" | "indefinite"

Table 19 – Common attributes of animation elements

Attribute	DataType	Default value	Description
repeatDur	<Clock-value> / "indefinite"	/	Specifies the total duration for repeat. The possible values: "<Clock-value>" | "indefinite".
fill	<string>	Remove	Indicates the animation effect at the last value of the active duration .The possible values: "freeze" | "remove".
from	<value>	/	Specifies the starting value of the animation. The value must be legal for the specified target attribute.
To	<value>	/	Specifies the ending value of the animation. The value must be legal for the specified target attribute.
by	<value>	/	Specifies a relative offset value for the animation. The value must be legal for the specified target attribute.

7.7.2 The 'animate' element

The 'animate' element is used to animate a single attribute or property over time.

A typical example of 'animate' element is shown in Figure 9.

```
<svg width="320" height="240" xmlns="http://www.w3.org/2000/svg">
   <g>
      <text font-family="arial" font-size="40" y="135" x="120" fill="green">
      Welcome!
      <animate attributeName="x" from="120" to="30" begin="0s" dur="3s"
               repeatCount="indefinite" />
      </text>
   </g>
   <rect fill="none" stroke="gray" x="1" y="1" width="318" height="238"/>
</svg>
```

Welcome!
Welcome!

(a) At zero seconds (b) At three seconds

Figure 9 – Example for the "animate" element

7.7.3 The 'set' element

The 'set' element provides a simple means of setting the value of an attribute for a specified duration.

A typical example of 'set' element is shown in Figure 10.

```
<svg width="320" height="240" xmlns="http://www.w3.org/2000/svg">
    <rect fill="none" stroke="gray" x="1" y="1" width="318" height="238"/>
    <rect fill="#3366FF" x="60" y="60" width="200" height="120">
        <set attributeName="fill" from="#3366FF" to="#00CCCC" begin="2s" />
    </rect>
</svg>
```

(a) At zero seconds
(b) At two seconds

Figure 10 – Example for the "set" element

7.8 Reference

SVG uses XLink for all links definitions, which should be supported in IPTV services.

7.8.1 IRI reference attributes

IRI references are normally specified with an 'href' attribute in the XLink namespace. The following XLink attributes are defined to provide information regarding the referenced resource.

Table 20 – IRI reference attributes

Attribute	DataType	Default value	Description
xlink:href	<IRI>	empty string	The location of the referenced object, expressed as an IRI reference.
xlink:type	<string>	simple	Identifies the type of XLink being used. In SVG Tiny 1.2, only simple links are available.
xlink:role	<IRI>	empty string	An optional IRI reference that identifies some resource that describes the intended property.
xlink:arcrole	<IRI>	empty string	An optional IRI reference that identifies some resource that describes the intended property.
xlink:title	<string>	/	Describes the meaning of a link or resource in a human-readable fashion, along the same lines as the role or arcrole attribute.

In all cases, an explicit XLink namespace declaration must be provided whenever one of the above attributes is used within an SVG content. One simple way to provide the declaration is to include an 'xmlns' attribute for the XLink namespace on the 'svg' element for content that uses XLink attributes.

The following example shows XLink namespace declaration and an image referencing in SVG document.

```
 <?xml version="1.0"?>
 <svg xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/svg"
 version="1.2" baseProfile="tiny">
   <desc> XLink namespace declaration and external image referencing</desc>
   <image xlink:href="extenalImage.png"/>
</svg>
```

7.8.2 Reference restriction

Some SVG elements using IRI references have restrictions on them. Reference restriction can be categorized as the following four types:

A) Reference a fragment within the current SVG document fragment (e.g., "#myelement").

B) Reference a fragment of an external SVG document (e.g., "afile.svg#anelement").

C) Reference an SVG document (e.g., "afile.svg").

D) Reference a media resource other than SVG, with or without the use fragments (e.g., "myimage.jpg" or "mycontainer#fragment").

Table 21 indicates whether the given attribute is allowed to have a reference with regard to the above four types. An IRI reference that does not comply with the restrictions is an invalid IRI reference.

Table 21 – Reference restriction on elements and attributes

Elements	Referencing attribute	A	B	C	D
'set'	xlink:href	Yes	No	No	No
'animate'	xlink:href	Yes	No	No	No
'a'	xlink:href xlink:role xlink:arcrole	Yes	Yes	Yes	Yes
'use'	xlink:href	Yes	Yes	No	No
'image'	xlink:href	No	No	No	Yes

NOTE – A 'use' element in an IRI reference type A is prohibited from referencing a 'svg' element. In type B, itis prohibited from referencing the fragment containing scripting, hyperlinking to animations or any externally referenced 'use' or 'animation' elements. An 'image' element is required to reference an image of raster format. An 'audio' or 'video' element is required to reference the supported audio or video formats.

7.8.3 Linking out of SVG content

SVG provides an 'a' element, analogous to HTML's 'a' element, to indicate links (also known as hyperlinks or web links). Each simple link associates exactly two resources, one local and one remote, with an arc going from the former to the latter. A simple link is defined for each separate rendered element contained within the 'a' element. For each rendered element within an 'a' element, the given rendered element is the local resource (the source anchor for the link). The remote resource (the destination for the link) is defined by an IRI specified by the XLink href attribute on the 'a' element. The remote resource may be any web resource (e.g., an image, a video clip, a sound bite, a program, another SVG document, an HTML document, etc.). By activating these links (by clicking with the mouse, through keyboard input, voice commands, etc.), users may traverse hyperlinks to these resources.

A typical example of the 'a' element is as follows. In this example, it assigns a link to an ellipse and clicking on the ellipse will cause the current window or frame to be replaced by the W3C home page.

```
 <?xml version="1.0"?>
 <svg width="5cm" height="3cm" version="1.2" baseProfile="tiny"
      xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
   <desc>Example link01 - a link on an ellipse
   </desc>
   <rect x=".01" y=".01" width="4.98" height="2.98"fill="none" stroke="blue"
        stroke-width=".03"/>
   <a xlink:href="http://www.w3.org">
     <ellipse cx="2.5" cy="1.5" rx="2" ry="1" fill="red" />
   </a>
</svg>
```

7.8.4 Linking into SVG content

As SVG content often represents a picture or drawing of something, a common need is to link into a particular view of the document, where a view indicates the initial transformations so as to present a closeup of a particular section of the document.

To link into a particular view of an SVG document, the IRI fragment identifier must be a correctly formed SVG fragment identifier. An SVG fragment identifier defines the meaning of the "selector" or "fragment identifier" portion of IRIs that locates resources of MIME media type "image/svg+xml".

An SVG fragment identifier can come in two forms:

1) Shorthand bare name form of addressing (e.g., MyDrawing.svg#MyView). This form of addressing, which allows addressing an SVG element by its ID, is compatible with the fragment addressing mechanism for older versions of HTML and the shorthand bare name formulation in "XPointer Framework" [b-W3C XPTRFW].

2) SVG view specification (e.g., MyDrawing.svg#svgView(transform(scale(2))). This form of addressing specifies the desired view of the document (e.g., the region of the document to view, the initial zoom level) completely within the SVG fragment specification. The contents of the SVG view specification is transform (...) whose parameters have the same meaning as the corresponding attribute has on a 'g' element).

An SVG fragment identifier is defined as follows:

```
SVGFragmentIdentifier ::= BareName |SVGViewSpec
BareName ::= XML_Name
SVGViewSpec ::= 'svgView(' SVGViewAttributes ')'
SVGViewAttributes ::= SVGViewAttribute |SVGViewAttribute ';' SVGViewAttributes
SVGViewAttribute ::= transformSpec
transformSpec ::= 'transform(' TransformParams ')'
```

where:

1) XML_Name conforms to the XML production for 'XML Name'

2) TransformParams corresponds to the parameter values for the transform attribute that is available on many elements. For example, transform(scale(5)).

8 Advanced profile

Since the IPTV TD full-fledged model [ITU-T H.722] supports not only basic IPTV services like linear TV and video-on-demand (VoD), but also supplementary IPTV services like Internet, medical, communication and etc., the enhanced capabilities' requirement of SVG's modules, elements, attributes and properties, which is also a subset of [W3C SVG Tiny], is defined to be supported in this section as "Advanced Profile". The IPTV TD full-fledged model, which chooses SVG as a multimedia application framework engine, is recommended to support SVG advanced profile.

8.1 Document structure

8.1.1 The 'svg' element

In addition to the attributes defined in basic profile, the attributes of 'svg' element defined in the advanced profile may also include the attributes described in Table 22.

Table 22 – Extended attributes of the 'svg' element

Attribute	DataType	Default value	Description
viewBox	<list-of-numbers> / "none"	none	Specifies a rectangular region that child graphical content must stretch to fit. The possible values is "<list-of-numbers>" | "none".
preserveAspectRatio	<string>	xMidYMid meet	It is available for all elements that establish a new viewport (see elements that establish viewports), indicates whether or not to force uniform scaling. For detailed information, please refer to [W3C SVG Tiny].
playbackOrder	<string>	all	Indicates whether it is possible to seek backwards in the document. The possible values are: "forwardOnly" | "all".
timelineBegin	<string>	onLoad	Controls the initialization of the timeline for the document. The possible values are: "onLoad" | "onStart".
zoomAndPan	<string>	magnify	Specifies whether the user agent provides any magnification and panning controls to the 'svg' element. The possible values are: "magnify" | "disable".

8.1.2 The 'g' element

Besides the attributes defined in basic profile, the attributes of 'g' element defined in the advanced profile may also include the attributes in Table 23.

Table 23 – Extended attributes of the 'g' element

Attribute	DataType	Default value	Description
transform	<transform-list> / <transform-ref> / "none"	None	Specifies a coordinate system transformation or nested transformations to apply to the element it appears on. The possible values are "<transform-list>" | "<transform-ref>" | "none". It supports the following values of <transform-list> in Advanced Profile: translate(<tx> [<ty>]): specifies a translation by tx and ty. scale(<sx> [<sy>]: specifies a scale operation by sx and sy. matrix(<a> <b> <c> <d> <e> <f>): specifies a transformation in the form of a transformation matrix

Table 23 – Extended attributes of the 'g' element

Attribute	DataType	Default value	Description
			of six values. rotate(<rotate-angle> [<cx> <cy>]): specifies a rotation by <rotate-angle> degrees about a given point. skewX(<skew-angle>): specifies a skew transformation along the x-axis. skewY(<skew-angle>): specifies a skew transformation along the y-axis. It also supports simple constrained transformations by using the 'ref(...)' value on this attribute in Advanced Profile. For detailed information, please refer to clause 7.6 of [W3C SVG Tiny].

8.1.3 The 'discard' element

The 'discard' element allows authors to specify the time at which particular elements are to be discarded, thereby reducing the resources required by a user agent in compliance with this Recommendation. This is particularly useful to help the resource restrained IPTV TDs conserve memory while displaying long-running documents. The attribute of 'discard' element for IPTV service is described in Table 24.

Table 24 – Attributes of the 'discard' element

Attribute	DataType	Default value	Description
xlink:href	<IRI>	empty string	An IRI reference that identifies the target element to discard.
begin	<begin-value-list>	0s	Indicates when the target element will be discarded. See the definition of 'begin' on animation elements for details.

8.1.4 The 'use' element

Besides the attributes defined in basic profile, the attributes of 'use' element defined in the advanced profile may also include the extended attributes listed in Table 25.

A typical example of the "use" element is shown in Figure 11.

Table 25 – Extended attributes of the 'use' element

Attribute	DataType	Default value	Description
transform	<transform-list> / <transform-ref> / "none"	none	See the definition of transform in clause 8.1.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     xmlns:xlink=http://www.w3.org/1999/xlink
     width="10cm" height="3cm" viewBox="0 0 100 30">
  <desc>'use' with nested transformations</desc>

<defs>
    <rect id="MyRect" x=".2" y=".2" width="30" height="10" fill="none"
 stroke="red"/>
  </defs>

  <rect x=".1" y=".1" width="99.8" height="29.8"
        fill="none" stroke="blue" stroke-width=".2"/>
  <use xlink:href="#MyRect" transform="scale(2) skewX(30)"/>
  </svg>
```

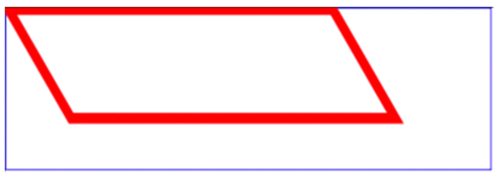


Figure 11 – Example for the "use" element

8.1.5 The 'image' element

Besides the attributes defined in basic profile, the attributes of 'use' element defined in the advanced profile may also include the attributed described in Table 26.

Table 26 – Extended attributes of the 'image' element

Attribute	DataType	Default value	Description
preserveAspectRatio	<string>	xMidYMid meet	It indicates whether elements shall establish a new viewport and force uniform scaling. It only applies when a value has been provided for 'viewBox' on the same element. For detailed information, please refer to [W3C SVG Tiny].

8.1.6 The 'switch' element

SVG provides a 'switch' element and five conditional processing attributes ('requiredExtensions', 'requiredFeatures', 'requiredFonts', 'requiredFormats' and 'systemLanguage'), as shown in Table 27, that provide the ability to specify alternate content depending on the capabilities of a given SVG user agent or the user's language.

The 'switch' element is a container element that can be used to select one of its child elements to process based on their conditional processing attributes.

Table 27 – Attributes of the 'switch' element

Attribute	DataType	Default value	Description
requiredExtensions	<list-of-string>	/	A conditional processing attribute that controls conditional processing based on whether the specified extensions are supported by IPTV TDs. The value is a list of IRI references which identify the required extensions.
requiredFeatures	<list-of-string>	/	A conditional processing attribute that controls conditional processing based on whether the specified features are supported by the SVG user agent. The value is a list of feature strings.
requiredFonts	<list-of-string>	/	A conditional processing attribute that controls conditional processing based on whether the specified fonts are available. The value is a list of font family names.
requiredFormats	<list-of-string>	/	A conditional processing attribute that controls conditional processing based on whether the media types [b-IETF RFC 2046] are supported by the SVG user agent. The value is a list of Internet media types supported by IPTV TDs.
systemLanguage	<list-of-string>	/	A conditional processing attribute that controls conditional processing based on the system language codes [ISO 639-1]. The value is a comma-separated list of language tags.

8.1.7 The 'prefetch' element

The 'prefetch' element provides a hint to the SVG user agent that part or all of the referenced media should be fetched ahead of time to make document playback smoother.

The attributes of this element are given in Table 28.

Table 28 – Attributes of the "prefetch" element

Attribute	DataType	Default value	Description
mediaSize	<long>	empty string	Defines how much of the media to fetch in bytes in terms of the file size of the media.
mediaTime	<clock-value>	/	Defines how much of the media to fetch in terms of the duration of the media. For discrete media (such as image), it causes the entire resource to be prefetched. If neither mediaSize nor mediaTime is specified, the whole resource shall be prefetched.
bandwidth	<long>	/	Defines how much network bandwidth to perform the prefetch.
mediaCharacterEncoding	<string>	/	Indicates the XML character set encoding (UTF-8, ISO-8859-1, etc.) that the 'mediaSize' attribute applies to.

Table 28 – Attributes of the "prefetch" element

Attribute	DataType	Default value	Description
mediaContentEncodings	<list-of-strings>	/	Indicates a white space separated list of the content encodings (as defined in HTTP/1.1 [b-IETF RFC 2616]) that were applied to encode the data.
xlink:href	<IRI>	empty string	An IRI reference to the resource to prefetch.

A typical example of 'prefetch' element is as follows. In this example, the external image is prefetched to ensure smooth animation, although it is not required to be rendered in the beginning of 10 seconds.

```
<desc>
  Prefetch the large image before starting the animation if possible.
</desc>

<defs>
  <prefetch xlink:href="http://www.example.com/images/huge1.png"/>
</defs>
<image x="0" y="0" width="400" height="300"
       xlink:href="http://www.example.com/images/huge1.png" display="none">
  <set attributeName="display" to="inline" begin="10s"/>
  <animate attributeName="xlink:href"
        values="http://www.example.com/images/huge1.png" begin="15s" dur="30s"/>
</image>
```

8.2 Fonts

8.2.1 The 'font-face-src' element

For Full-fledged model of IPTV TD, 'font-face-src' element together with 'font-face-uri' element is used to display the style of the character by external font sources.

8.2.2 The 'font-face-uri' element

The 'font-face-uri' element is used within a 'font-face-src' element to reference a font defined inside or outside of the current SVG document.

The attributes of this element are given in Table 29.

Table 29 – Attributes of the "font-face-uri" element

Extended attribute	DataType	Default value	Description
xlink:href	<IRI>	empty string	An IRI reference to the external source

The following is a typical example of the "font-face-src" and "font-face-uri" element.

```
<font xml:id="Font1">
  <font-face font-family="sans-serif" font-weight="bold" font-style="normal" >
    <font-face-src>
      <font-face-uri xlink:href="font01.svg#la"/>
    </font-face-src>
  </font-face>
</font>
<text x="100" y="100" font-family="sans-serif" font-weight="bold"
      font-style="normal" font-size="55">Text using embedded font
</text>
```

8.3 Shape

Shape elements can use the common attribute 'transform', 'focusable' and Navigation. See the definition of transform in clause 8.1.2 and the definition of Navigation in clause 7.6.3.

8.3.1 The 'ellipse' element

The 'ellipse' element defines an ellipse which is axis-aligned with the current user coordinate system based on a centre point and two radii.

The attributes of this element are given in Table 30.

A typical example of the "ellipse" element is shown in Figure 12.

Table 30 – Attributes of the "ellipse" element

Attribute	DataType	Default value	Description
cx	<coordinate>	0	The x-axis coordinate of the centre of the eclipse, e.g., x="100".
cy	<coordinate>	0	The y-axis coordinate of the centre of the eclipse, e.g., y="100".
rx	<length>	0	The x-axis radius of the ellipse, e.g., rx="10". The value "0" indicates the rendering of the element is disabled.
ry	<length>	0	The y-axis radius of the ellipse, e.g., ry="10". The value "0" indicates the rendering of the element is disabled.

```
< ellipse cx="100" cy="100" rx="20" ry="10" stroke="#000066" fill="#FFFF99"/>
```

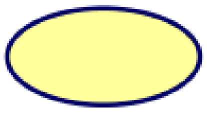


Figure 12 – Example for the "ellipse" element

8.3.2 The 'path' element

The 'path' element represents the geometry of the outline of an object, defined in terms of *moveto* (set a new current point - M), *lineto* (draw a straight line - L), *curveto* (draw a curve using a cubic Bézier - C) and *closepath* (close the current shape by drawing a line to the last *moveto* - Z) instructions. For detailed information, please refer to clause 8.3 of [W3C SVG Tiny]. See also Table 31.

A typical example of the 'path' element is shown in Figure 13.

Table 31 – Attributes of the "path" element

Attribute	DataType	Default value	Description
d	<string>	empty string	The definition of the outline of a shape. It is described in combination of command and coordinate(s). See also the definition of command in clause 8.3 of [W3C SVG Tiny]. E.g., d="M 100 100 L 300 100 L 200 300 Z". The value empty string "" indicates the rendering of the element is disabled.
pathLength	<number>	0	The authoring length of the path, e.g., pathLength="10". The value "0" indicates no authoring length is specified.
NOTE – In the IPTV service, only absolute coordinate is used for the "d" attribute's command and the "user unit" for the "pathLength" attribute indicates that one unit of the video output resolution of the IPTV TD.			

```
<path d="M 100 100 L 300 100 L 200 200 Z" pathLength="648" fill="#00FF00" />
```

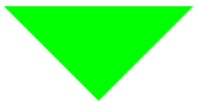


Figure 13 – Example for the "path" element

8.4 Text

8.4.1 The 'text' element

Besides the attributes defined in the basic profile, the attributes of 'text' element defined in the advanced profile may also include the attributes shown in Table 32.

Table 32 – Extended attributes of the "text" element

Attribute	DataType	Default value	Description
transform	<transform-list> / <transform-ref> / "none"	None	See the definition of transform in clause 8.1.2.

8.4.2 The 'textArea' element

The 'textArea' element allows simplistic wrapping of text content within a given region.

The attributes of this element are given in Table 33.

A typical example of the 'textArea' element is shown in Figure 14.

Table 33 – Attributes of the "textArea" element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The x-axis coordinate of one corner of the rectangular region into which the text content will be placed.
y	<coordinate>	0	The y-axis coordinate of one corner of the rectangular region into which the text content will be placed.
width	<coordinate>	auto	The width of the rectangular region into which the text content will be placed. A value of 'auto' indicates that the width of the rectangular region is infinite.
height	<coordinate>	auto	The height of the rectangular region into which the text content will be placed. A value of 'auto' indicates that the height of the rectangular region is infinite.
editable	<string>	none	Indicates whether the text can be edited. Possible values are: none: editing facilities are not provided for the content. simple: user agent must provide a way to edit the content which is not hidden or removed from the rendering tree.
focusable	<string>	auto	Defines if an element can get keyboard focus and be a target for field-to-field navigation actions. Possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
transform	<transform-list> / <transform-ref> / "none"	none	See the definition of transform in clause 8.1.2.
NOTE – If both 'width' and 'height' have the value 'auto', the text will be rendered in a single line along the direction of the text progression until all the text is rendered, or until a line-breaking element is encountered. Any text which does not fit the given rectangular region will not be rendered.			

```
<textArea font-size="25" font-family="Georgia" x="10" y="10" width="200"
          height="300">
  Get a 7-day free trial of VoD service to see what you've been missing. After
  your trial period, you'll get VoD service for $8.99/month subscription.
</textArea>
<rect x="5" y="5" width="210" height="310" stroke-width="3" stroke="#777"
      fill="none"/>
```

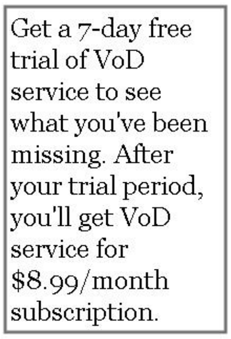

Get a 7-day free trial of VoD service to see what you've been missing. After your trial period, you'll get VoD service for $8.99/month subscription.

Figure 14 – Example for the "textArea" element

8.4.3 The 'tspan' element

The 'tspan' element adjusts graphic and font properties within a text content element except positional attributes such as 'x' and 'y'.

The attributes of this element are given in Table 34.

A typical example of the 'tspan' element is shown in Figure 15.

Table 34 – Attributes of the "tspan" element

Attribute	DataType	Default value	Description
focusable	<string>	auto	Defines if an element can get keyboard focus and be a target for field-to-field navigation actions. Possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.

```
<rect x="60" y="100" width="300" height="90" fill="none" stroke="blue" />
<text x="100" y="150" fill="blue" font-size="20">
  Link to
   <tspan font-weight="bold" fill="red" font-size="24">the favourite</tspan>
   Movie.
</text>
```

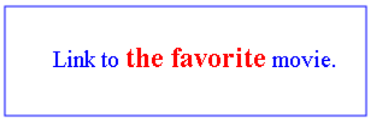


Figure 15 – Example of the "tspan" element

8.5 Painting

8.5.1 The 'linearGradient' element

The 'linearGradient' element is used to paint in a linear colour transitions along a vector. It is not rendered directly but referenced using the 'fill' and 'stroke' properties.

The attributes of this element are given in Table 35.

Table 35 – Attributes of the "linearGradient" element

Attribute	DataType	Default value	Description
x1	<coordinate>	0	Defines the starting point in x-axis coordinate onto which the gradient stops shall be mapped.
y1	<coordinate>	0	Defines the starting point in y-axis coordinate onto which the gradient stops shall be mapped.
x2	<coordinate>	1	Defines the ending point in x-axis coordinate onto which the gradient stops shall be mapped.
y2	<coordinate>	0	Defines the ending point in y-axis coordinate onto which the gradient stops shall be mapped.
gradientUnits	<string>	objectBoundingBox	Defines the coordinate system for attributes 'x1', 'y1', 'x2', 'y2' that shall be used when rendering the gradient. The possible values are: "userSpaceOnUse" | "objectBoundingBox".
NOTE – If 'x1' = 'x2' and 'y1' = 'y2', then the area to be painted shall be painted as a single colour using the colour and opacity of the last gradient stop. If the gradient starts or ends inside the bounds of the target rectangle, the terminal colours of the gradient shall be used to fill the remainder of the target region.			

8.5.2 The 'radialGradient' element

The 'radialGradient' element is used to paint in a given radial gradient. It is not rendered directly but referenced using the 'fill' and 'stroke' properties.

The attributes of this element are given in Table 36.

Table 36 – Attributes of the "radialGradient" element

Attribute	DataType	Default value	Description
cx	<coordinate>	0.5	Define the x-axis coordinate of the centre of the element.
cy	<coordinate>	0.5	Define the y-axis coordinate of the centre of the element.
r	<coordinate>	0.5	Define the radius of the largest circle for the radial gradient.
gradientUnits	<string>	objectBoundingBox	Defines the coordinate system for attributes 'cx', 'cy', 'r' that shall be used when rendering the gradient. The possible values are: "userSpaceOnUse" | "objectBoundingBox".

8.5.3 The 'stop' element

The 'stop' element is used to stop the colour ramp on a gradient. It is the child element of the 'linearGradient' element or the 'radialGradient' element.

This element applies 'stop-color' property and 'stop-opacity' property to specify the colour and opacity at the gradient stop. See clause 9.2.

The attributes of this element are given in Table 37.

A typical example of 'stop' element is shown in Figure 16.

Table 37 – Attributes of the "stop" element

Attribute	DataType	Default value	Description
offset	<number>	0	Indicates where the gradient stop shall be placed. For linear gradients, the 'offset' attribute represents a location along the gradient vector. For radial gradients, it represents a relative distance from ('cx', 'cy') to the edge of the outermost/largest circle.

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     width="800" height="400">
  <g>
    <defs>
      <linearGradient xml:id="MyGradient1" gradientUnits="userSpaceOnUse"
         x1="50" y1="100" x2="350" y2="300">
        <stop offset="0.05" stop-color="#F60"/>
        <stop offset="0.95" stop-color="#FF6"/>
      </linearGradient>
      <radialGradient xml:id="MyGradient2" gradientUnits="userSpaceOnUse"
                      cx="600" cy="200" r="150">
        <stop offset="0" stop-color="red"/>
        <stop offset="0.5" stop-color="blue"/>
        <stop offset="1" stop-color="red"/>
      </radialGradient>
    </defs>
    <rect fill="none" stroke="blue" x="1" y="1" width="798" height="398"/>
    <!-- The rectangle is filled using a linear gradient paint server -->
    <rect fill="url(#MyGradient1)" stroke="black" stroke-width="5"
          x="50" y="100" width="300" height="200"/>
   <!-- The rectangle is filled using a radial gradient paint server -->
    <rect fill="url(#MyGradient2)" stroke="black" stroke-width="5"
          x="450" y="100" width="300" height="200"/>
  </g>
</svg>
```

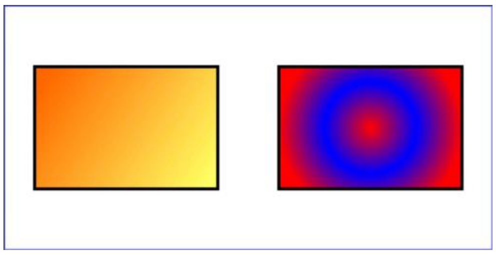


Figure 16 – Example for the "stop" element

8.6 Interactivity

Besides Interactivity functionality defined in basic profile, SVG content can also be interactive by utilizing the following features:

– User-initiated actions such as a key-press can cause timed elements (such as animation and multimedia elements in the advanced profile) to start or stop, scripts to execute or 'listener' elements to trigger 'handler' elements.

– Users are able to zoom into and pan around SVG content depending on the value of the 'zoomAndPan' attribute on the 'svg' element.

SVG user agents that operate in interaction-capable user environments are required to support the ability to magnify and pan. Magnification represents a complete, uniform transformation on an SVG document fragment, where the magnify operation scales all graphical elements by the same amount. Panning represents a translation (i.e., a shift) transformation on an SVG document fragment in response to a user interface action.

8.7 Multimedia

Multimedia contains elements and attributes to describe a timed sequence of media objects. Media elements define their own timelines within their time container.

Multimedia elements can apply 'audio-level' property to control the volume of a particular audio or video. See clause 9.2.

8.7.1 The common attributes

Multimedia supports the SVG timing and runtime synchronization attributes. They are used to control the timing and runtime synchronization of the timed elements, respectively.

Table 38 – Common attributes of Multimedia elements

Attribute	DataType	Default value	Description
Timing Attributes	<value-list>	/	Controls the timing of the media, including 'begin', 'dur', 'end', 'min', 'max', 'restart', 'repeatCount', 'repeatDur', 'fill'. See clause 7.7.1.
syncBehavior	<string>	default	Defines the runtime synchronization behaviour for an element. The possible values are: "canSlip" | "locked" | "independent" | "default". The default value is determined by the value of the syncBehaviorDefault attribute specified on the 'svg' element. For detailed information, please refer to clause 12.6 [W3C SVG Tiny].
syncTolerance	<Clock-value>" / "default"	default	Defines the synchronization tolerance for the associated element. The possible values are: "<Clock-value>" | "default". The default value is determined by the value of the syncToleranceDefault attribute specified on the 'svg' element.
syncMaster	<boolean>	false	Boolean attribute on media elements and time containers that forces other elements in the time container to synchronize their playback to this element. The possible values are: "true" | "false".

8.7.2 The 'audio' element

The 'audio' element specifies an audio file which is to be rendered to provide synchronized audio.

The attributes of this element are given in Table 39.

A typical example of the 'audio' element is shown in Figure 17. In this example, the audio will be played three times when the button is clicked.

Table 39 – Attributes of the "audio" element

Attribute	DataType	Default value	Description
xlink:href	<IRI>	empty string	An IRI reference used to link to the audio content.
type	<content-type>	/	The audio format. For optimizing download time by requiring a particular content format, it is suggested to use 'requiredFormats' instead of 'type'.

```
<audio xlink:href="ouch.ogg" audio-level="0.7" type="application/ogg"
        begin="mybutton.click" repeatCount="3"/>
  <g xml:id="mybutton">
    <rect width="150" height="50" x="20" y="20" rx="10"
      fill="#ffd" stroke="#933" stroke-width="5"/>
    <text x="95" y="55" text-anchor="middle" font-size="30"
      fill="#933">Play Audio</text>
        </g>
```

Figure 17 – Example for the "audio" element

8.7.3 The 'video' element

The 'video' element specifies a video file which is to be rendered to provide synchronized video.

The attributes of this element are given in Table 40.

Table 40 – Attributes of the "video" element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The x-axis coordinate of the rectangular region into which the video is placed. If the transform behaviour of the video is geometric, this coordinate is one corner of the rectangular region. If it is pinned, this coordinate is the pin point of the rectangular region.
y	<coordinate>	0	The y-axis coordinate of the rectangular region into which the video is placed. If the transform behaviour of the video is geometric, this coordinate is one corner of the rectangular region. If it is pinned, this coordinate is the pin point of the rectangular region.
width	<length>	0	The width of the rectangular region into which the video is placed. A negative value shall be treated as unsupported. If the transform behaviour of the video is geometric, a value of zero shall disable rendering of the element. If it is pinned, this attribute shall have no effect on rendering.

Table 40 – Attributes of the "video" element

Attribute	DataType	Default value	Description
height	<length>	0	The height of the rectangular region into which the video is placed. A negative value shall be treated as unsupported. If the transform behaviour of the video is geometric, a value of zero shall disable rendering of the element. If it is pinned, this attribute shall have no effect on rendering.
xlink:href	<IRI>	empty string	An IRI reference used to link to the video content.
type	<content-type>	/	The video format. For optimizing download time by requiring a particular content format, it is suggested to use 'requiredFormats' instead of 'type'.
Overlay	<string>	none	Defines whether a 'video' is rendered according to the SVG painter's model or on top of all other SVG elements. The attribute value can be either of the following: top: a temporary video canvas must be set aside and the 'video' is drawn last in the whole document's compositing process. none: the 'video' must be rendered according to the SVG painter's model.
initalVisibility	<string>	whenStarted	Controls the visibility of the media object before its first active duration period has started. The attribute value can be either of the following: whenStarted: indicates that the media object is not displayed until its first active duration starts. always: the media element is visible from the initialization of the parent time container.
focusable	<string>	auto	Defines if an element can get keyboard focus and be a target for field-to-field navigation actions. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.
preserve AspectRatio	<string>	xMidYMid meet	Indicates whether or not to force uniform scaling. For detailed information, please refer to [W3C SVG Tiny].
transform Behavior	<string>	geometric	Defines whether a video is transformed/resampled or pinned/unresampled. The possible values are: "geometric" | "pinned" | "pinned90" | "pinned180" | "pinned270". For detailed information, please refer to clause 12.3.1 of [W3C SVG Tiny].

A typical example of the 'video' element is as follows. In this example, the timeline of the document and of the video are independent. The video begins as soon as the video element is parsed. When the video starts, it will start from the first frame.

```
<?xml version="1.0"?>
<svg xml:id="A" xmlns="http://www.w3.org/2000/svg" baseProfile="tiny"
     xmlns:xlink="http://www.w3.org/1999/xlink" version="1.2"
     timelineBegin="onStart">          <!-- process time = t0 -->
<!-- ...[many elements]... -->         <!-- additional process time = t1 = 5s -->
  <video xlink:href="myvideo.avi" audio-level="0.8" type ="video/x-msvideo"
        width="320" height="240" x="50" y="50" begin="0s"
        syncBehavior="independent"/>    <!-- additional process time = t2 = 1s -->
</svg>
```

8.7.4 The 'animation' element

The 'animation' element specifies an SVG document providing synchronized animated vector graphics. It does not support pointing to document fragments within SVG files.

The attributes of this element are given in Table 7-41.

Table 41 – Attributes of the "animation" element

Attribute	DataType	Default value	Description
x	<coordinate>	0	The x-axis coordinate of the rectangular region into which the video is placed.
y	<coordinate>	0	The y-axis coordinate of the rectangular region into which the video is placed.
width	<length>	0	The width of the rectangular region into which the video is placed. A negative value shall be treated as unsupported.
height	<length>	0	The height of the rectangular region into which the video is placed. A negative value shall be treated as unsupported.
xlink:href	<IRI>	empty string	An IRI reference to the animation data.
initalVisibility	<string>	whenStarted	Controls the visibility of the media object before its first active duration period has started. See the definition of initalVisibility in clause 8.7.3.
preserveAspectRatio	<string>	xMidYMid meet	Indicates whether or not to force uniform scaling. For detailed information, please refer to [W3C SVG Tiny].
focusable	<string>	auto	Defines if an element can get keyboard focus and be a target for field-to-field navigation actions. The possible values are: "true" | "false" | "auto".
Navigation Attributes	<string>	auto	See definition of Navigation in clause 7.6.3.

A typical example of the 'animation' element is as follows. This animation involves another SVG document. The effect is: the animation begins at 1s and has 3s of duration, repeats 1.5 times and then freezes.

```
<animation begin="1" dur="3" repeatCount="1.5" fill="freeze"
           x="100" y="100" width="50" height="50"
           xlink:href="bouncingBall.svg"/>
```

8.8 Animation

8.8.1 The common attributes

Besides the common attributes defined in basic profile, the following attributes shall be supported in advanced profile.

Table 42 – Extended common attributes of Animation elements

Attribute	DataType	Default value	Description
calcMode	<string>	linear	Specifies the interpolation mode for the animation. The possible values are: "discrete" | "linear" | "paced" | "spline".
values	<list>	/	Specifies a semicolon-separated list of one or more values in which the animation is applied in order. If this attribute is specified, any 'from', 'to' or 'by' attribute values are ignored.
keyTimes	<list>	/	Defines a semicolon-separated list of time values used to control the pacing of the animation.
keySplines	<list>	/	Defines a set of Bézier control points associated with the 'keyTimes' list to controls interval pacing.
additive	<string>	replace	Controls whether or not the animation is additive. The possible values are: "replace" | "sum".
accumulate	<string>	none	Controls whether or not the animation is cumulative. The possible values are: "none" | "sum".

8.8.2 The "animateColor" element

The "animateColor" element specifies a colour transition over time. It uses the same calculation method as an "animate" element is animating a colour value.

8.8.3 The "animateMotion" element

The "animateMotion" element causes its target element to move along a motion path.

The attributes of this element are given in Table 43.

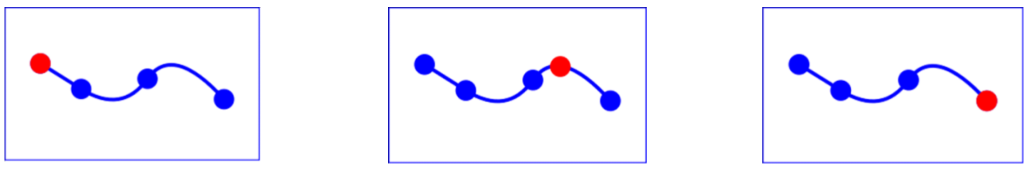
A typical example of "animateMotion" element is shown in Figure 18.

Table 43 – Properties of the "animateMotion" element

Attribute	DataType	Default value	Description
path	<path-data>	empty string	The motion path, expressed in the same format and interpreted the same way as the 'd' attribute on the 'path' element.
keyPoints	<list-of-numbers>	/	Defines a semicolon-separated list of floating point values between 0 and 1 to indicate how far along the motion path the target element shall move at the moment in time specified by corresponding 'keyTimes' value.
rotate	<string>	0	Post-multiplies a supplemental transformation matrix onto the current transformation matrix of the target element to apply a rotation transformation. The possible values are: "auto" | "auto-reverse" | "<number>".
NOTE – The 'calcMode' attribute used in 'animateMotion' element has the default value of 'paced', which will produce constant velocity motion along the specified path. For usage or constraint of animation common attributes, please refer to clause 16.2.14 of [W3C SVG Tiny].			

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink" version="1.2" baseProfile="tiny"
    width="5cm" height="3cm" viewBox="0 0 500 300" >
  <desc> Example demonstrating the use of the 'animationMotion' element.</desc>

  <rect x="1" y="1" width="498" height="298"
        fill="none" stroke="blue" stroke-width="2"/>
  <path id="path1" d="M70,110 T150,160 T280,140 T430,180"
       fill="none" stroke="blue" stroke-width="7.06"/>
  <circle cx="70" cy="110" r="20" fill="blue"/>
  <circle cx="150" cy="160" r="20" fill="blue"/>
  <circle cx="280" cy="140" r="20" fill="blue"/>
  <circle cx="430" cy="180" r="20" fill="blue"/>
  <circle r="20" fill="red">
    <animateMotion path="M70,110 T150,160 T280,140 T430,180"
     begin="0s" dur="4s" repeatCount="indefinite" rotate="auto"
     calcMode="linear" fill="freeze">
   </animateMotion>
  </circle>
</svg>
```



(a) At zero seconds (b) At three seconds (c) At four seconds

Figure 18 – Example for the "animateMotion" element

8.8.4 The 'mpath' element

The 'mpath' element is used to reference an existing 'path' element for use as a motion animation path. It may appear only as the child of an 'animateMotion' element to which it provides the path.

A typical example of 'mpath' element is shown in Figure 19.

The attributes of this element are given in Table 44.

Table 44 – Properties of the "mpath" element

Attribute	DataType	Default value	Description
xlink:href	<IRI>	empty string	An IRI reference to the 'path' element that defines the motion path.

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink" version="1.2" baseProfile="tiny"
    width="5cm" height="3cm" viewBox="0 0 500 300" >
  <desc> Example demonstrating the use of the 'mpath' element.</desc>

  <rect x="1" y="1" width="498" height="298"
       fill="none" stroke="blue" stroke-width="2"/>
 <path id="path1" d="M70,110 T150,160 T280,140 T430,180"
       fill="none" stroke="blue" stroke-width="7.06"/>
 <circle r="20" fill="red">
     <animateMotion begin="0s" dur="4s" repeatCount="indefinite" rotate="auto"
         calcMode="linear" fill="freeze">
       <mpath xlink:href="#path1"/>
     </animateMotion>
  </circle>
</svg>
```

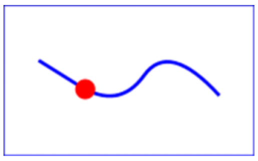


Figure 19 – Example for the "mpath" element (at 1s)

8.8.5 The 'animateTransform' element

The 'animateTransform' element animates a transformation attribute on a target element, thereby allowing animations to control translation, scaling, rotation and/or skewing.

The attributes of this element are given in Table 45.

A typical example of 'animateTransform' element is shown in Figure 20.

Table 45 – Properties of the "animateTransform" element

Attribute	DataType	Default value	Description
type	<string>	/	Indicates the type of transformation which is to have its values change over time. The possible values are: "translate" | "scale" | "rotate" | "skewX" | "skewY".

```
<svg width="320" height="320" xmlns="http://www.w3.org/2000/svg">
    <rect fill="none" stroke="gray" x="1" y="1" width="318" height="238"/>
    <g>
       <text font-family="Arial" font-size="50" y="90" x="30"
      fill="blue" >IPTV</text>
       <animateTransform attributeName="transform" begin="0s" dur="3s"
             type="scale" from="1" to="2" repeatCount="indefinite"/>
     </g>
</svg>
```

Figure 20-a – At zero seconds

Figure 20-b – At three seconds

Figure 20 – Example for the "animateTransform" element

8.9 Reference

8.9.1 Reference restriction

Besides the reference restriction defined in basic profile, additional element and attribute in the table below is allowed to have a reference.

NOTE – A 'font-face-uri' element is required to reference a SVG 'font' element. An 'mpath' element is required to reference a 'path' element.

Table 46 – Reference restriction on additional elements and attribute

Elements	Referencing attribute	A	B	C	D
'discard'	xlink:href	Yes	No	No	No
'prefetch'	xlink:href	Yes	Yes	Yes	Yes
'font-face-uri'	xlink:href	Yes	Yes	No	No
'mpath'	xlink:href	Yes	No	No	No
'animateColor'	xlink:href	Yes	No	No	No
'animateMotion'	xlink:href	Yes	No	No	No
'animateTransform'	xlink:href	Yes	No	No	No
'audio'	xlink:href	No	No	No	Yes
'video'	xlink:href	No	No	No	Yes
animation	xlink:href	No	No	Yes	No

9 Styling

SVG uses styling properties to describe many of its document parameters. Styling properties define how the graphics elements in the SVG content are to be rendered. SVG shares some of its styling properties with CSS [b-W3C CSS2] and XSL [b-W3C XSL]. The normative definition of these sharing properties should conform to the definition from the CSS 2 specification [b-W3C CSS2]. Meanwhile, SVG specification defines many styling properties that should conform to SVG-specific rules.

The following properties are shared between CSS 2 and SVG:

– 'font-family'
– 'font-size'
– 'font-style'
– 'font-weight'
– 'display'
– 'visibility'

9.1 Styling usage in IPTV services

SVG Tiny 1.2 does not require support for CSS selectors applied to SVG content. SVG user agents must not rely on external or author stylesheets to style documents. SVG supports the way of specifying styling properties by the corresponding presentation attributes. For each styling property defined in clause 9.2, there is a corresponding presentation attribute with the same name that can be applied to all relevant SVG elements. Property declarations via presentation attributes are expressed in XML, which is case-sensitive and must match the exact property name.

SVG styling may be applied to IPTV services in the following usage scenarios:

– SVG content used as an exchange format (style sheet language-independent): Since support for a particular style sheet language is not guaranteed across all implementations, it is a requirement that SVG content can be fully specified without the use of a style sheet language.

– SVG content generated as the output from XSLT: XSLT can be used to transform XML data extracted for instance from databases into an SVG graphical representation of that data. It is a requirement that fully specified SVG content can be generated from XSLT [b-W3C XSLT].

9.2 Styling properties for IPTV services

SVG Profiles for IPTV services support the SVG styling properties shown in Table 47.

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
font-family	<string>	system's default font	Fonts	Basic profile / Advanced profile	See clause 7.2.4.
font-style	<string>	normal	Fonts	Basic profile / Advanced profile	See clause 7.2.4.
font-weight	<string>	normal	Fonts	Basic profile / Advanced profile	See clause 7.2.4.
font-size	<string>	medium	Fonts	Basic profile / Advanced profile	It refers to the size of the font between baseline to baseline. Percentage values are not supported. Absolute length unit is suggested.
fill	<paint>	black	Painting	Basic profile / Advanced profile	It specifies the interior of the given graphical element must be painted. The possible values are: "none" | "currentColor" | "<color>" | "<FuncIRI>" | "<system paint>" | "inherit".

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
fill-opacity	<opacity-value> / "inherit"	1	Painting	Basic profile / Advanced profile	It specifies the opacity of the painting operation which shall be used to paint the interior of the current object. The possible values are: "<opacity-value>" | "inherit".
stroke	<paint>	none	Painting	Basic profile / Advanced profile	It specifies the outline of the given graphical element must be painted. The possible values are: "none" | "currentColor" | "<color>" | "<FuncIRI>" | "<system paint>" | "inherit".
stroke-width	<length> / "inherit"	1	Painting	Basic profile / Advanced profile	It specifies the width of the stroke which shall be used on the current object. The possible values are: "<length>" | "inherit".
stroke-linecap	<string>	butt	Painting	Basic profile / Advanced profile	It specifies the shape which shall be used at the end of open subpaths when they are stroked. The possible values are: "butt" | "round" | "square" | "inherit".
stroke-opacity	<opacity-value> / "inherit"	1	Painting	Basic profile / Advanced profile	It specifies the opacity of the painting operation used to stroke the current object. The possible values are: "<opacity-value>" | "inherit".
viewport-fill	<paint>	none	Painting	Basic profile / Advanced profile	It specifies the colour which shall be used to fill the viewport created by a particular element. The possible values are: "none" | "currentColor" | "<color>" | "inherit".
viewport-fill-opacity	<opacity-value> / "inherit"	1.0	Painting	Basic profile / Advanced profile	It specifies the opacity of the 'viewport-fill' that shall be used for a particular element. The possible values are: "<opacity-value>" | "inherit".

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
solid-color	<paint>	black	Painting	Basic profile / Advanced profile	It specifies the colour that shall be used for a given element. The possible values are: 'currentColor' | '<color>' | 'inherit'.
solid-opacity	<opacity-value> / "inherit"	1	Painting	Basic profile / Advanced profile	It defines the opacity of a given element. The possible values are: '<opacity-value> ' | 'inherit'.
display	<string>	inline	Painting	Basic profile / Advanced profile	It controls the rendering of graphics elements or container elements. It doesn't affect the object's existence in the DOM, but that in the rendering tree. For detailed information, please refer to [W3C SVG Tiny].
visibility	<string>	visible	Painting	Basic profile / Advanced profile	It controls the rendering of graphics elements on the canvas. It doesn't affect the object's existence either in the DOM or in the rendering tree. The possible values are: "visible" | "hidden" | "collapse" | " inherit".
opacity	<opacity-value> / "inherit"	1	Painting	Basic profile / Advanced profile	It specifies the opacity of the 'image' element. The possible values are: "<opacity-value>" | "inherit".
text-anchor	<string>	start	Text	Advanced profile	It is used to align a string of text relative to a given point. Possible values are "start" | "middle" | "end".
display-align	<string>	auto	Text	Advanced profile	It specifies the alignment, in the block-progression-direction, of the text content of the 'textArea' element. The possible values are "auto" | "before" | "center" | "after".
line-increment	<string>	auto	Text	Advanced profile	It provides limited control over the size of each line in the block-progression-direction. The possible values are "auto" | "number".

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
audio-level	<number>	1.0	Multimedia	Advanced profile	Specifies a value that is used to calculate the volume of a particular audio. Values below 1.0 decrease it and a value of zero silences it.
fill-rule	<string>	nonzero	Painting	Advanced profile	It indicates the algorithm which must be used to determine what parts of the canvas are included inside the shape. The possible values are: "nonzero" | "evenodd" | "inherit".
stroke-linejoin	<string>	miter	Painting	Advanced profile	It specifies the shape which shall be used at the corners of shapes when they are stroked. The possible values are: "miter" | "round" | "bevel" | "inherit".
stroke-miterlimit	<miterlimit> / "inherit"	4	Painting	Advanced profile	It imposes a limit on the ratio of the miter length to the 'stroke-width' when miter joins are specified for 'stroke-linejoin'. When the limit is exceeded, the join must be converted from a miter to a bevel. The possible values are: "<miterlimit>" | "inherit".
stroke-dasharray	"none" / <list-of-lengths> / "inherit"	none	Painting	Advanced profile	It specifies the pattern of dashes and gaps that shall be used to stroke paths. The possible values are: "none" | "<list-of-lengths>" | "inherit".
stroke-dashoffset	< length> / "inherit"	0	Painting	Advanced profile	It specifies the distance into the dash pattern that must be used to start the dash. The possible values are: "< length>" | "inherit".
vector-effect	<string>	none	Painting	Advanced profile	It determines whether the outline of an object keeps its original when transforms are applied. The possible values are: "non-scaling-stroke" | "none" | "inherit".

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
stop-color	<paint>	black	Painting	Advanced profile	It specifies the colour that shall be used for the gradient "stop". The possible values are: "currentColor" | "<color>" | "inherit".
stop-opacity	<opacity-value> / "inherit"	1	Painting	Advanced profile	It specifies the opacity that shall be used for the gradient "stop". The possible values are: "<opacity-value>" | "inherit".
color-rendering	<string>	auto	Painting	Basic profile / Advanced profile	It provides a hint to the implementation about how to make speed versus quality tradeoffs as it performs colour interpolation and compositing. The possible values are: "auto" | "optimizeSpeed" | "optimizeQuality" | "inherit".
shape-rendering	<string>	auto	Painting	Basic profile / Advanced profile	It provides a hint to the implementation about what tradeoffs to make as it renders vector graphics elements. The possible values are: "auto" | "optimizeSpeed" | "crispEdges" | "geometricPrecision" | "inherit".
text-rendering	<string>	auto	Painting	Basic profile / Advanced profile	It provides a hint to the implementation about what tradeoffs to make as it renders text. The possible values are: "auto" | "optimizeSpeed" | "optimizeLegibility" | "geometricPrecision" | "inherit".

Table 47 – Styling properties for IPTV

Property	DataType	Default value	Applicable on Module	Applicable on profile	Description
image-rendering	<string>	auto	Painting	Basic profile / Advanced profile	It provides a hint to the implementation about how to make speed versus quality tradeoffs as it performs image processing. The possible values are: "auto" | "optimizeSpeed" | "optimizeQuality" | "inherit".
buffered-rendering	<string>	auto	Painting	Basic profile / Advanced profile	It provides a hint to the implementation about how often an element is modified to make speed versus memory tradeoffs as it performs rendering. The possible values are: "auto" | "dynamic" | "static" | "inherit".
pointer-events	<string>	all	Interactivity	Basic profile / Advanced profile	Specifies under what circumstances a given graphics element can be the target element for a pointer event. The possible values are: "boundingBox" | "visiblePainted" | "visibleFill" | "visibleStroke" | "visible" | "painted" | "fill" | "stroke" | "all" | "none" | "inherit".

10 Pagination

SVG Print is a set of guidelines to produce final-form documents in XML suitable for archiving and printing.

Annex A

Basic data types

(This annex forms an integral part of this Recommendation.)

Annex A defines a number of basic data types used in the definitions of SVG properties and attributes.

Table A.1 – Data type of SVG properties and attributes

Attribute	Description
<boolean>	A boolean value, specified as either 'true' or 'false'.
<char>	A character, as defined by the Char production in [b-W3C XML 1.0].
<clock-value>	An amount of time, used by various attributes on timed elements.
<color>	The basic type <color> defines a colour within the sRGB colour space.
<content-type>	An Internet media type, as defined in [b-IETF RFC 2046].
<coordinate>	A <coordinate> is a length in the user coordinate system that is the given distance from the origin of the user coordinate system along the relevant axis (the x-axis for X coordinates, the y-axis for Y coordinates).
<focus>	The type of value that can be used in the various navigation attributes, such as 'nav-next', 'nav-prev', etc.
<font-family-value>	A list of font family names and generic names.
<family-name>	A single font family name as given by a <family-name>.
<font-size-value>	A value that can be used for the 'font-size' property, excluding the 'inherit' value.
<ID>	The type of value that can be used in an XML attribute of type ID.
<integer>	An <integer> is specified as an optional sign character ("+" or "−") followed by one or more digits "0" to "9".
<IRI>	An internationalized resource identifier.
<language-id>	The type of value accepted by the 'xml:lang' attribute.
<length>	A length is a distance measurement.
<list-of-content-types>	A space-separated list of Internet media types.
<list-of-strings>	A <list-of-strings> consists of a separated sequence of <string>s.
<list-of-*T*s>	(Where *T* is a type other than <content-type>, <string>, <language-id> and <family-name>.) A list consists of a separated sequence of values.
<string>	A sequence of zero or more <Char>s.
<number>	A <number> value is specified in either decimal or scientific notation.
<QName>	A qualified name as defined by the QName production in Namespaces in XML 1.0 or XML 1.1.
<long>	An optional sign character ("+" or "−") followed by one or more digits "0" to "9".
<path-data>	The type is used to represent path data.
<paint>	The type of paint to use when filling or stroking a given graphics element.
<FuncIRI>	Functional notation for an IRI: "url(" <IRI> ")".
<XML-NMTOKEN>	An XML name token, as defined by the Nmtoken production in Extensible Markup Language (XML) 1.0 or 1.1.
<IDREF>	The type of value that can be used in an XML attribute of type IDREF, that is a string matching the Name production in Extensible Markup Language (XML) 1.0 or 1.1.

Annex B

Overview of uDOM

(This annex forms an integral part of this Recommendation.)

This Annex gives an overview of SVG uDOM.

B.1 Supported methods

SVG uDOM supports methods for the following:

– Document access

  The SVG uDOM offers access to a Document object which is the root for accessing other features.

– Tree navigation

  SVG uDOM only allows navigation of the element nodes in the DOM tree.

– Element creation

  SVG uDOM allows creation of new elements.

– Element insertion

  SVG uDOM allows the insertion of an element.

– Element removal

  SVG uDOM allows removal of elements.

– Attribute and property eccess

  SVGT 1.2 uDOM supports two ways of accessing XML attributes and CSS properties; the standard way via getAttributeNS and setAttributeNS on the Element interface and via a new concept called Traits.

– Event Listener Registration and Removal

  SVG uDOM allows adding and removing EventListeners.

– Animation

  SVG uDOM allows code to start or end timed elements.

B.2 uDOM modules

a) Module: dom
   - i) DOMException
   - ii) Node
   - iii) Element
   - iv) Document
   - v) DOMImplementation

b) Module: events
   - i) EventTarget
   - ii) EventListener
   - iii) Event
   - iv) MouseEvent
   - v) WheelEvent

vi) TextEvent

vii) KeyboardEvent

viii) UIEvent

ix) ProgressEvent

x) ConnectionEvent

c) Module: smil

i) ElementTimeControl

d) Module: global

i) Global

ii) GlobalException

iii) Connection

iv) Timer

e) Module: svg

i) SVGException

ii) SVGDocument

iii) SVGElementInstance

iv) SVGSVGElement

v) SVGRGBColor

vi) SVGRect

vii) SVGPoint

viii) SVGPath

ix) SVGMatrix

x) SVGLocatable

xi) SVGLocatableElement

xii) TraitAccess

xiii) Additional accessing rules

xiv) ElementTraversal

xv) SVGElement

xvi) SVGTimedElement

xvii) SVGVisualMediaElement

xviii) EventListenerInitializer2

xix) TimeEvent

xx) SVGGlobal

xxi) AsyncStatusCallback

xxii) AsyncURLStatus.

Appendix I

SVG elements for IPTV TD models

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

ITU-T terminal devices Recommendations define IPTV TD basic model [ITU-T H.721], IPTV TD mobile model [ITU-T H.723] and IPTV TD full-fledged model [ITU-T H.722]. The full-fledged IPTV terminal device can support the provision of IPTV services with more improved capabilities, features, compared with basic model IPTV terminal devices. According to the capability requirements of different TD models, SVG elements and attributes will be classified into core part for IPTV, which is implemented by the basic model and the mobile model, and an extended part for IPTV, which is implemented by full-fledged model. The relationship of SVG elements for the various TD models is illustrated in Figure I.1.

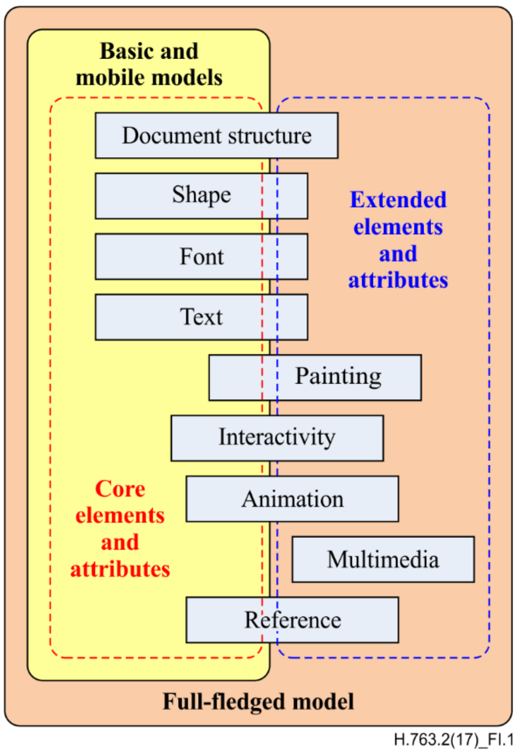


Figure I.1 – SVG element for IPTV TD models

NOTE – The frame within the red dashed lines represents the scope of SVG core elements and attributes, which should be supported by IPTV TD basic model and mobile model. The frame within the blue dashed lines represents the scope of SVG extended elements and attributes, which should be supported by the IPTV TD full-fledged model instead of the basic model and the mobile model. The yellow filled frame represents the scope of the SVG basic profile corresponding to the IPTV TD basic model and mobile model. The orange filled frame represents the scope of the SVG advanced profile corresponding to IPTV TD full-fledged model.

SVG element can be divided into nine functional modules (grey boxes in Figure I.1) according to element characteristics. These functional modules are partially or fully supported by different IPTV TDs. A brief description is as follows:

1) Document structure: Document structure elements are used to describe and identify a standard composition of SVG document structure;

2) Shape: SVG shape elements are used to describe the basic and complex shape of the composition;

3) Font: To create SVG content with arbitrary fonts and know that the same graphical result will appear when the content is viewed by all end users, even when end users do not have the necessary fonts installed on their computers;

4) Text: SVG text elements are used to describe the way the text is to be rendered;

5) Painting: SVG elements of the painting are used to describe shapes, text and other elements of the manifestations and effects;

6) Interactivity: SVG interactivity elements are used to describe the event detection and response methods, and the script used to trigger events;

7) Animation: SVG document fragments can describe time-based modifications to the document's elements. Using the various animation elements can define motion paths, fade-in or fade-out effects, and objects that grow, shrink, spin or change colour;

8) Multimedia: SVG multimedia elements are used to describe audio and video multimedia download, play method;

9) Reference: Reference element and attributes are used to describe the various ways for the SVG element reference, including external references and internal references.

These elements or part of the elements' attributes are classified into core part and extended part, depending on whether the element is indispensable or its rendering takes too much of TD's resources.

Bibliography

[b-IETF RFC 2046]	IETF RFC 2046 (2013), *Multipart Internet Mail Extensions (MIME) Part Two: Media Types*.
[b-IETF RFC 2616]	IETF RFC 2616 (2009), *Hypertext Transfer Protocol.*
[b-IETF RFC 3987]	IETF RFC 2046 (2015), *Internationalized Resource Identifiers*.
[b-W3C CSS2]	W3C Recommendation CSS2 (2007), *Cascading Style Sheets, level 2*. <https://www.w3.org/TR/2007/CR-CSS21-20070719/>
[b-W3C DOM2]	W3C Recommendation DOM2 (2000): *Document Object Model (DOM) Level 2 Core Specification.* <https://www.w3.org/TR/DOM-Level-2-Core/>
[b-W3C DOM3]	W3C Recommendation DOM3 (2004), *Document Object Model (DOM) Level 3 Core Specification*. <https://www.w3.org/TR/DOM-Level-3-Core/>
[b-W3C SMIL]	W3C Recommendation SMIL 2.1 (2005), *Synchronized Multimedia Integration Language.* <https://www.w3.org/TR/SMIL2/>
[b-W3C SVG]	W3C Recommendation 16 August 2011, *Scalable Vector Graphics (SVG) 1.1 (Second Edition).* <https://www.w3.org/TR/SVG>
[b-W3C XML 1.0]	W3C Recommendation XML 1.0 (2008), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. <https://www.w3.org/TR/REC-xml/>
[b-W3C XPTRFW]	W3C Recommendation 25 March 2003, *XPointer Framework*. <https://www.w3.org/TR/xptr-framework/>
[b-W3C XSL]	W3C Recommendation XSL (2006), *Extensible Stylesheet Language (XSL) Version 1.1*. <https://www.w3.org/TR/xsl/>
[b-W3C XSLT]	W3C Recommendation XSLT (2010), *XML Linking Language (XLink) Version 1.1*. <https://www.w3.org/TR/xlink11/>

Printed in Switzerland
Geneva, 2017