SMPTE 397M-2003

SMPTE STANDARD

for Television —
Declarative Data Essence — Transitional

Contents

Approved
July 9, 2003

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Administrative practices.

SMPTE 397M-2003 was prepared by Technology Committee on Data Essence Technology.

Declarative Data Essence — Transitional

1 Scope

This document defines a standard for the authoring of declarative data content intended to be combined primarily with video and/or audio services, and distributed to data-capable television signal receivers. The data essence defined in this standard may be used in conjunction with that defined in the Declarative Data Essence – Content Level 1 specification [DDE-1]. It will be possible for applications authored according to the essence specified herein to execute on the vast majority of currently fielded interactive digital television systems, either directly or after being transcoded. Declarative content is generally nonprocedural, and most commonly in the form of HTML. However, procedural scripting is also defined.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

[CSS-TV] CR-css-tv-20020807, W3C Candidate Recommendation: CSS TV Profile 1.0 (August 8th, 2002).

[DOM-0] SMPTE 366M-2002, Television — Declarative Data Essence — Document Object Model Level 0 (DOM-0) and Related Object Environment.

[DOM2-Core] REC-DOM-Level-2-Core-20001113, W3C Recommendation: Document Object Model (DOM) Level 2 Core Specification Version 1.0 (November, 13th, 2000).

[DOM2-Style] REC-DOM-Level-2-Style-20001113, W3C Recommendation: Document Object Model (DOM) Level 2 Style Specification Version 1.0 (November 13th, 2000).

[DOM2-HTML] REC-DOM-Level-2-HTML-20030109, W3C Recommendation: Document Object Model (DOM) Level 2 HTML Specification Version 1.0 (January 9th, 2003).

[DOM2-Events] REC-DOM-Level-2-Events-20001113, W3C Recommendation: Document Object Model (DOM) Level 2 Events Specification Version 1.0 (November 13th, 2000).

[ECMA-262] ECMA-262, ECMA Standard: ECMAScript 3rd edition (December 1999).

[ECMA-327] ECMA-327, ECMA Standard: ECMAScript 3rd edition, Compact Profile (June 2001).

[HTML-4.01] REC-html401-19991224, W3C Recommendation: HTML 4.01 Specification (December 24th, 1999).

[HTTP-1.1] RFC2616, IETF: Hypertext Transfer Protocol -- HTTP/1.1 (June 1999).

[HTTPS] RFC2818, IETF: HTTP Over TLS (May 2000).

[ITU-R BT.601-5] ITU-R BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios (10/95).

[JPEG] ISO/IEC 10918-1, Digital compression encoding of continuous-tone still images (JPEG), 1992.

[MPEG2-Audio] ISO/IEC 13818-3, 2nd edition, Information technology – Generic coding of moving pictures and associated audio – Part 3: MPEG-2 Audio, (1998).

[MPEG2-Video] ISO/IEC 13818-2 Information technology – Generic coding of moving pictures and associated audio information – Part 2: MPEG-2 Video, (1996).

[PNG] EC-png, W3C Recommendation: PNG (Portable Network Graphics) Specification Version 1.0 (October 1996).

[URI-TV] RFC2838, IETF: Uniform Resource Identifiers for Television Broadcasts (May 2000).

3 Glossary of acronyms, terms and data types

This section defines conformance keywords, acronyms, abbreviations, and terminology that apply to the specification in its entirety and to each constituent part.

3.1 Conformance keywords

As used in this document, the conformance keyword *shall* denotes a mandatory provision of the standard. The keyword *should* denotes a provision that is recommended but not mandatory. The keyword *may* denotes a feature whose presence does not preclude compliance, which may or may not be present at the option of the application implementer.

3.2 Acronyms and abbreviations

clut – color lookup table

CSS – Cascading Style Sheets

DOM -- Document Object Model

DVB – Digital Video Broadcasting

ECMA – European Computer Manufacturers Association

HTML – Hyper Text Mark-up Language

HTTP – Hyper Text Transport Protocol

JPEG – Joint Picture Expert Group

MHP – Multimedia Home Platform

MPEG – Motion Picture Expert Group

PNG – Portable Network Graphics

UA – User Agent

UI – User Interface

URL – Uniform Resource Locator

3.3 Terms

events – An instantiation of asynchronous notifications from a client platform to the interactive television content.

pel – A pixel in the broadcast video coordinate space.

service – A sequence of programs under the control of a broadcaster which can be broadcast as part of a schedule.

stream – A unidirectional continuous flow of content. Example: MPEG2 video.

tuning – The act of switching between two MPEG transport streams or multiplexes.

user agent – Any process that interprets HTML documents.

viewer's preferences – Choices that a viewer has made concerning which of several options for a particular functionality is to be set as a default; for example, choice of audio language or whether to use a keyboard or a remote control. Control over viewer's preferences is outside the scope of this specification because it is expected that receiver-specific applications would be used to set up the viewer's preferences and that these defaults would not be set by an interactive television application.

4 Introduction

As stated above, the goal of this specification is to document declarative data essence that will allow authors to design interactive tv content supportable by the vast majority of currently fielded interactive digital television systems.

4.1 Rendering model (informative)

This section is informative in its entirety.

4.1.1 Expected minimal client computing capabilities

Even though this specification is a content authoring specification and not a transcoding or receiver specification, it is useful for authors to understand some restrictions of the typical environment in which their applications will, at least initially, execute. Content authored according to this specification can be used to control the concurrent presentation of graphics and broadcast video. The content may be transcoded prior to dissemination to receivers because many of the receivers for which content is authored will have extremely limited memory and processing capabilities. Therefore, it is expected that some of the rendering, and the layout, may be done at a server in order to achieve adequate client performance. These limitations, and, hence, the need to perform some of the rendering at a server, explains why certain dynamic functionality is more restricted in this specification than in W3C specifications, for example.

4.1.2 Presentation model

The author using the essence described in this specification can designate content for both the video plane, which may render either apparently changing or constant/still video, and the graphics plane. For example, the author may select the service from which the video is to be displayed and may control the size and location of the video on the screen. The graphics plane logically lies on top of the video plane and, where supportable, transparency of content in the graphics plane with respect to the video underneath can be specified by the author.

Typically the video is viewed full screen and the content author can use this specification to indicate the size, location, and contents of a rectangle in the graphics plane that overlays the full screen video. However, this specification also allows the author to indicate graphics that occupy the entire screen and to specify the size, location, and contents of a video rectangle that appears, due to appropriate transparency, to be above or inset into the graphics plane. Even in this case, though, the video may be rendered in a plane logically below the graphics plane, with a transparent rectangle logically above it. Therefore, using the model with the video logically below the graphics plane, the content author can construct either what appears to be graphics on top of underlying full-screen video or a video window on top of underlying full-screen graphics.

4.1.3 Color and transparency

Besides memory and cpu restrictions, the majority of currently fielded receivers that are expected to be the first clients to present the content authored according to this specification typically have restricted color capabilities. Most often they cannot support true color models, but instead have support, in hardware, for the use of color palettes. With a palette-based model, the color of a pixel of an image or corresponding to characters of text is typically represented by an index into a color palette. However, specification of a palette index is not supported in HTML and is not proposed in this document, though a facility for providing a palette is supported. According to this specification, the author specifies a color by designating the color itself, rather than by designating an index into a palette. However, this specification provides a way for an author to indicate the entire palette. This model was chosen because it is better than both (1) requiring an author to specify the palette index and (2) disallowing the author from specifying a palette at all. Obviously without specially designed tools, option (1) is not very user friendly. Allowing the author to specify a palette which is in a form easily extractable via existing tools does allow an author to easily check that they have not used more colors than can be supported on a particular set of destination clients. If the clients are expected to be heterogeneous, or content has been authored to be interoperable on multiple heterogeneous networks, then both multiple palettes and alternative colors may be supplied, in conformance with CSS cascading rules. This specification allows a different palette to be specified per top level page, however, it is expected that clients that are limited so much that they rely on a hardware palette are also severely limited in other ways so that such clients would also not support an application model that would allow multiple top level pages to attempt to control the display simultaneously. Therefore, even though life cycle is beyond the scope of this specification (see below), it is expected that life cycle considerations in a given network will account for receiver hardware limitations.

In addition to restrictions on the number of total colors that are supportable on a given client, often the transparency models supported by a client are very limited. One of the most commonly supported models in digital receivers is that of allowing only a single transparent or semi-transparent color in the palette. On clients where there is only a single transparent or semi-transparent color available, it is useful for the content author to be able to specify the size and location of the contents of the graphics plane to be within a given rectangle. When so specified on such clients, the area outside the rectangle is not considered part of the graphics contents and therefore no color, and hence no transparency level, is associated with that area. Therefore, in very low end receivers, the only transparent or semi-transparent color will be available for use within the content of the specified rectangle within the graphics plane.

Another common transparency model is one that allows each palette element to have one of 2 values for their transparency attribute (often called the alpha channel); the first value being fully opaque and the second value being a palette-specific fixed transparency level.

As with color, the author can target their page at a particular transparency model or may supply transparency information for several models, allowing a user agent or transcoder to choose a model that the client supports.

4.2 Application model, life cycle, and security (informative)

This specification recognizes that application model, life cycle, and security are most often dictated by entities external to the author such as broadcasters and multi-system operators, or may be specified on a regional basis. Such operators may choose to use signalling as defined in other specifications [DVB-HTML, OCAP, BML] or may use their own proprietary signalling. Therefore, in order for authors in different regions of the world to be able to define interoperable applications using declarative data essence, this specification defines mechanisms for the author to use to control only presentation and viewer interaction. The only assumption concerning life cycle made in the definition of the data essence to be used by the author is that only a single application uses the screen resources at a time.

4.3 Organization of the specification

This specification is divided according to the class of data essence. Section 5 details the HTML markup available to the author, section 6 details the CSS style capabilities, section 7 defines the ECMAScript support expected, section 8 defines the IDL interfaces for the DOM capabilities and bindings from those interfaces to ECMAScript; section 9 defines media types available to the author, and section 10 defines the locator formats available to the author. Each of the sections is divided according to the following organization.

- Each of the sections relies heavily upon a basis derived from existing standards from the following groups: W3C, ECMA, SMPTE, and IETF. In those cases where entire specifications from those organizations are adopted, they are normatively referenced in their entirety.

- After the description of the primary specification or specifications upon which a section is based, a description of additional uses of the syntax and semantics from those references, if any, is presented. For example, the style section describes how the clip property from CSS2 [CSS2R1], when applied to a top level `div` element, can be used to improve performance and presentation.

- The next subsection describes portions of this specification that originate in other specifications, most notably various interactive television specifications such as DVB-HTML[DVB-HTML] and DASE[DASE-1].

- The last subsection of each section describes syntax and associated semantics introduced by this specification. This new syntax and semantics were designed in order to provide data essence content authors with functionality commonly employed by content authors using procedural-only interfaces on currently fielded systems, but which functionality has been missing from previously specified data essence specifications.

Where appropriate, many definitions, whether originating here or in a previous specification, are preceded by an informative introduction placing the definition in context.

5 HTML markup

5.1 HTML 4.01/XHTML 1.0

5.1.1 Compliance with HTML 4.01

Authors may provide either XHTML 1.0 [XHTML-1.0] or HTML 4.01 [HTML-4.01] content containing any of the elements below.

This specification includes all elements and attributes specified in HTML 4.01. As stated in Section A.3.1 of HTML 4.01, several HTML 3.2 elements (e.g., isindex) are deprecated.

5.2 HTML 4.01 syntax with additional semantics

5.2.1 Placing video within an object

5.2.1.1 Introduction to placing video within an object (informative)

Although the video plane is logically behind the graphics plane, content authors often want the entire video content to appear to be scaled to a rectangular region with graphics presented around and possibly overlapping the video. In the next subsection, this specification indicates how the author can achieve this effect by supplying an appropriate URI for certain attributes of particular elements. (For placement of a rectangular subset of the entire video content, see Section 6.3.1.)

5.2.1.2 Using tv: to place video within an object

A “tv:” URI, as described in section 10, can be used as either the `src` attribute of an `img` element or as the `data` attribute of an `object` element. Using CSS absolute positioning for the `img` or `object` element allows the content author to position the video rectangle. Additionally, the author can specify appropriate values for the CSS width and height properties in order to scale the video element. For more details on sizing and positioning, see section 6.1.

5.2.2 Tuning and stream selection

5.2.2.1 Using tv: for tuning and stream selection

When the content author has a priori knowledge of the DNS name that corresponds to a given channel, the author can use HTML to cause tuning to that channel using the `href` attribute in the `A` element. For example, the following HTML

```
<A id="my_link" href="tv://example.com">Click Me</A>
```

allows the HTML document to present a link, my_link, which requests the tuner to select a possibly new service (see section 10.2). If the request is authorized and resolves to a valid service, then the HTML document will be unloaded and replaced with a TV media handler playing the default video and audio streams associated with the service.

URLs defined in section10.2, may be used in the following HTML element attributes in order to cause tuning or service selection: a.href, img.src, and object.data. These attributes may be dynamically changed using the DOM as stated in Section 8.1.2. Tuning and/or service selection can also be achieved via CSS (see Section 6.2.3.2).

5.2.3 The accesskey attribute

5.2.3.1 Additional values for the accesskey attribute

This specification supports the HTML `accesskey` attribute. In addition to the `accesskey` values permitted by HTML 4.01, values for this attribute may also be chosen from the key name space defined below (see section 8.2).

5.3 New values for HTML Markup

5.3.1 When to render

5.3.1.1 Introduction to when to render (informative)

Since video is often displayed behind or concurrently with graphics in interactive television, it is not always appropriate to display portions of the graphics as they become available. Sometimes the author does not want any graphics displayed until some other resource is available; for example, until the background video or audio has started. This section describes how the author can use the LINK and META elements to stipulate resources that must be available prior to displaying any graphics. Additionally, it describes how to specify which of several rendering policies to use with the graphics content. Finally, it also describes how the author can specify a timeout, after which the author would desire the graphics to be rendered, independent of whether all of the prerequisites have been obtained and regardless of the particular rendering policy that was requested.

5.3.1.2 Using the LINK element to specify prerequisites

When used with a LINK element and an `href` attribute, the "`rel`" attribute with a value of "`prerequisite`" is used to indicate that no rendering of the page is to occur prior to acquiring the resource specified by the `href` attribute, assuming that no timeout (see Section 5.3.1.4) has expired.

For example,

```
<LINK href="http://www.example.com/background.mpg" rel="prerequisite">
```

5.3.1.3 Using the META element to indicate the rendering policy

The author can use the `name/content` pair within a `META` element in the `head` of a document to indicate the desired rendering policy. When an author uses this feature, they should specify `name = 'dde-render-`

policy' and content should be one of 'progressive','layoutComplete' or 'loadComplete,' where these content values have the following meanings.

The progressive policy indicates that authors of compliant content can assume that rendering shall start as soon as the essential resources (those marked as prerequisites in the LINK element) have been acquired.

The layoutComplete rendering policy indicates that the rendered image shall not be displayed until the software has acquired sufficient information to determine the complete on-screen layout and has acquired those resources labelled as prerequisites.

The loadComplete rendering policy indicates that the graphics shall not be displayed until all resources that will be used for rendering the display have been downloaded. The only difference between the loadComplete rendering policy and labeling all resources as prerequisites is that in the first case, the OnLoad event will have been delivered to the appropriate handler, if any, prior to rendering, and hence may affect the rendered view.

If no policy is specified, then the author is, in essence, stipulating that a UA-dependent policy may be used.

5.3.1.4 Using the META element to indicate the rendering timeout

The author can use the name/content pair within a META element in the head of a document to indicate the rendering timeout. By specifying a value for the timeout, the author is indicating a period after which the UA may revert to UA-dependent behaviour, rather than adhering to the specified rendering policy. The author may specify this timeout period by using name = 'dde-render-timeout' and setting content equal to an integer value. This integer value is specified in terms of seconds. If no timeout is specified, then the author is, in essence, stipulating that a UA-dependent timeout may be used.

If the specified timeout occurs, and at least all of the prerequisite resources have been acquired, then the author may expect that the new page will display, independent of the specified rendering policy. If some of the prerequisite resources have not been acquired, then the author cannot be sure whether the user agent will continue to show the previous page, if any, or whether it will display an error message or attempt to display those resources that it has been able to acquire.

5.3.2 Static Prefetch hints

5.3.2.1 Using the LINK element to specify static hints for prefetching

The HTML 4.01 specification enumerates many recognized LINK types that can be used within the LINK element in the HEAD section of a document, e.g., Alternate, Stylesheet, and Glossary. This section introduces another LINK type: Prefetch. This LINK type is used by an author to identify resources that the author believes are highly likely to be used after the presentation of those items on the current page. That is, this LINK type is used to provide a hint to the client as to which resources it may wish to prefetch.

Example of use:

```
<LINK href="http://www.example.com/larger_img.jpg" rel="prefetch">
```

5.3.3 Multi-mode clients

5.3.3.1 Introduction to non-video mode (informative)

As stated previously, some clients are short on computing resources, such as memory. Often these clients can be placed in one of two modes, either (1) a mode that supports concurrent display of graphics in the graphics plane and rendering of video in the video plane or (2) a mode that dedicates all resources to graphic display. In the second mode, memory that would be devoted to holding video frames as they are rendered, and possibly re-sized, is instead devoted to rendering of graphics for the graphics plane. Also, on some very primitive clients, it is not possible to use both the video and the graphics plane concurrently; therefore, user agents on these clients may only render pages that indicate that their graphic plane content is not to be presented concurrently with video. Further, it is also common to have some virtual channels, e.g., a game channel or a weather channel, that

correspond only to content to be rendered in the graphics plane with no background video. Therefore, it is useful on low-end clients to be aware when such virtual channels are being displayed. In this case, these receivers need not use valuable resources for display of video. Hence, the next section shows how authors can indicate whether their pages are to be rendered with background video.

5.3.3.2 Specification of video mode

DDE-Transitional uses the `name/content` pairs "`requested/`*`feature-name`*" introduced by DASE [DASE-1], in the generic `META` element within the head element, to specify that video is not required to be presented with the content of the page. The particular *`feature-name`* introduced here is "`graphics-only`." Therefore, the statement below indicates that the author does not expect video to be displayed concurrently with the graphics:

```
<META name="requested" content="graphics-only"/>.
```

6 Style

Style for DDE-Transitional is based upon W3C's Cascading Style Sheets.

6.1 CSS-TV

All properties as identified in CSS TV Profile 1.0 [CSS-TV] are available to the author, with the exception of the following modifications. The use of the clut property as described in section 6.4.1.2 is mutually exclusive with the use of the opacity property. The clip property (included by reference in the CSS TV Profile 1.0), having semantics completely identical to that specified in the CSS2 revision 1 specification, provides the author with the ability to specify information that can be used to more efficiently present their content (as described in Section 6.2.3.1). As stated in the CSS TV Profile 1.0, "The inability of a TV-UA to implement part of this specification due to the limitations of a particular device SHALL NOT imply non-conformance." In this regard, authors need to be aware of two common limitations of current receivers. The first typical limitation includes restrictions on size and positioning. That is, although authors may carefully indicate, for example, that the video be scaled to a particular size and placed at a specific location, the receiver may only be capable of approximating the requested size and/or position. The second limitation has to do with restrictions on dynamic access to the style information used in a transcoding environment, particularly where there are substantial restrictions on the bandwidth available for the transcoded content. Dynamic access limitations are detailed in Section 8.1.3.

6.2 Additional uses of existing CSS

In this section, use of existing CSS properties, consistent with semantics described in CSS TV Profile, to achieve certain affects desirable in interactive television content authoring is described. Additionally, the dde-tv media type, employed in these uses as well as in CSS described in Sections 6.3 and 6.4, is described in this section.

6.2.1 Introduction to additional use of existing CSS properties (informative)

As stated in the introduction, this DDE-Transitional specification assumes that only a single application has control over the screen at a time, therefore an author defining DDE-Transitional content can assume that the canvas has exclusive access to the entire television screen.

6.2.2 Additional media type: dde-tv

A CSS media type is a collection of CSS style properties associated with a particular media form or a manner in which content is presented on a media form. A number of media types are pre-defied in CSS2 [CSS2]. This specification defines a new media type, `dde-tv`, to be used as the default media type for stylesheet content. All of the CSS properties identified below as well as the CSS TV Profile properties referred to in Section 6.1 above comprise the `dde-tv` media type.

6.2.3 Additional use of existing CSS properties

6.2.3.1 Defining the active graphics-plane area

As stated above, some clients support only a very limited transparency model (i.e., the client supports only a single transparent or semi-transparent color). Sometimes authors will target their content to such clients or to a heterogeneous mix of clients, some of whom support only this most limited transparency model. In such cases, the authors will have more flexibility if they can indicate the location and size of the graphics rectangle that they will be targeting. This additional flexibility means that they can obtain maximum utilization of the few transparent colors within the graphics. The author may indicate the size and location of the graphics rectangle by incorporating all information to be presented in the graphics plane within a top level 'div' element and applying the CSS2 'clip' property to that div element in order to specify the rectangular region within which the graphics are to appear. The 'clip' property, however, was not correctly specified in the 1998 version of the CSS, Level 2 specification and has since been corrected in the CSS, Level 2, Revision 1 document [CSS2R1]. However, as of the date of this document, that revision has not yet reached candidate status, and, as of this date, the CSS for TV specification does not reflect that correction; hence we include the definition of the clip property from the revised CSS, Level 2 specification below.

'clip'

Value: <shape> | auto | inherit

Initial: auto

Applies to: block level and replaced elements

Percentages: N/A

Inherited: no

Media: dde-tv

The 'clip' property applies only to absolutely positioned elements. Values have the following meanings:

auto

The clipping region has the same size and location as the element's box(es).

<shape>

The only valid <shape> value is: rect (<top>, <right>, <bottom>, <left>) where <top> and <bottom> specify offsets from the top border edge of the box, and <right> and <left> specify offsets from the left border edge of the box in left-to-right text and from the right border edge of the box in right-to-left text. Authors should separate offset values with commas. Authors may find that some user agents also support separation by white space, because the first version of the CSS 2 specification was ambiguous in this respect.

<top>, <right>, <bottom>, and <left> may either have a <length> value or 'auto'. Negative lengths are permitted. The value 'auto' means that a given edge of the clipping region will be the same as the edge of the element's generated box (i.e., 'auto' means the same as '0' for <top> and <left> (in left-to-right text, <right> in right-to-left text), the same as the computed value of the height plus the sum of vertical padding and border widths for <bottom>, and the same as the computed value of the width plus the sum of the horizontal padding and border widths for <right> (in left-to-right text, <left> in right-to-left text), such that four 'auto' values result in the clipping region being the same as the border edge).

The element's ancestors may also have clipping regions (e.g. if their 'overflow' property is not 'visible'); in this case, the author can expect that what is rendered is the intersection of the various clipping regions.

If the clipping region exceeds the bounds of the UA's document window, content may be clipped to that window by the native operating environment.

6.2.3.2 Background-image of the body

As stated in Section 5.2.2.1, tuning and/or stream selection can be accomplished by setting particular attributes in an HTML element. Additionally, the background-image property of the body can be set to a tv: locator as described in section 10.2 in order to specify video, if any, that is to be presented in the video plane which logically lies behind the graphics plane.

6.3 CSS Properties adapted from other specifications

6.3.1 Clipping and subsequent scaling of video

6.3.1.1 Introduction to clipping and subsequent scaling of video (Informative)

The ability to select a rectangular component of a broadcast video and subsequently scale and display the selected component is discussed in this section.

6.3.1.2 Positioning a rectangular portion of the video

This specification adopts the use of the “crop” property that is currently being considered by W3C for the CSS, level 3, box model [CSS3-box]. The “crop” property allows the author to select a rectangular component of any replaced element, which includes video. In order to subsequently scale the selected rectangular component, absolute positioning can be used. This “crop” property is a generalization of the “clip-video” property [DVB-HTML]. Since the CSS, level 3, box model is currently in working draft status, specification of the crop property is included herein.

```
dde-crop:
   Value: <shape> | auto
   Initial: auto
   Applies to: replaced elements
   Inherited: no
   Percentages: relative to intrinsic size
   Media: dde-tv
   Computed value: specified value
```

`'auto'`

The element's computed width and height are the same as its actual width and height.

The only shape that the author can assume will be supported is the rect(top, right, bottom, left) shape as defined next.

`rect(top, right, bottom, left)`

Each of the four arguments can be a `<length>` or a `<percentage>`. If the element to which this is applied is video, then the length values can be specified in ‘pels’, which is the pixel size associated with input video. All percentage values are computed relative to the intrinsic dimensions of the element. Values are offsets relative to the top left of the element. The computed intrinsic width and height of the element are determined by subtracting the left from the right for the width, and similarly top from bottom for the height. However, if this computation results in a negative value, it is considered to be zero.

6.3.2 Porter-Duff composition

6.3.2.1 Introduction to Porter-Duff composition property (informative)

Porter-Duff rules are typically used to indicate the type of transparency composition that is to be done when the area occupied by one object overlays the area occupied by another object. Most receivers are only capable of supporting the src rule, but it is useful to be able to specify a different rule for those receivers with sufficient processing power to support those other rules.

6.3.2.2 Porter-Duff composition property

The author can use the 'compose rule' property, which is adapted from DVB-MHP [DVB-HTML], to specify the Porter-Duff rule that they prefer to be used to blend objects in the graphics plane into the background video. The default rule is src-over; however, receivers lacking sufficient processing power to support the src-over rule may approximate it (or any of the other computationally intensive rules) by using the src rule. The meanings of the composition rule values are completely defined by Porter and Duff[Porter-Duff].

```
'dde-compose-rule'

   Value: clear | dst-in | dst-out | dst-over | src | src-in | src-out | src-over

   Initial: src-over

   Applies to: all elements

   Inherited: no

   Percentages: N/A

   Media: dde-tv
```

6.3.3 Navigation properties

6.3.3.1 Introduction to the navigation properties (informative)

BML [BML], DASE[DASE-1], and DVB-HTML [DVB-HTML] define navigation properties, though their definitions differ. As in DVB-HTML and DASE, the initial value is the default UA transition for the <nav-direction> properties, which means that these properties do not always explicitly have to be used in order to designate the location to which navigation will precede next when a particular arrow key is struck.

6.3.3.2 Navigation properties

The dde-nav-index property may be used to specify that an element is the target of a directional navigation property (see dde-nav-left, dde-nav-right, dde-nav-up, dde-nav-down properties).

```
'dde-nav-index'

   Values: auto | <number> | inherit

   Initial: auto

   Applies to: all focusable elements

   Inherited: no

   Percentages: n/a

   Media: dde-tv
```

When the value is `auto`, directional navigation behaviour is not defined by this specification, but is implementation dependent.

When a <number> value is used to express an integer index value, no more than one element in a document instance shall be associated with the index.

The `dde-nav-[left, right, up, down]` properties may be used to specify an element to which focus should be moved when responding to focus movement events.

The following constraints apply to these properties:

```
Values: auto | <number> | <string> | inherit

Initial: auto

Applies to: all focusable elements

Inherited: no

Percentages: n/a

Media: dde-tv
```

When the value is `auto`, directional navigation behaviour is not defined by this specification, but is implementation dependent.

When a `<number>` value form is used, it shall denote an integer index value that corresponds to the index value associated with an element by the `dde-nav-index` property.

When a `<string>` value form is used, it shall adhere to the following syntax:

```
navigation-target:  [frame-ref] '#' element-ref

frame-ref:       idref

element-ref:     idref | '$' integer
```

The value of `frame-ref`, if specified, shall be a valid identifier reference associated with some frame in the current document instance frame hierarchy. The value of `element-ref` shall be either a valid identifier reference to some element or an integer reference to an index value associated with an element by a `dde-nav-index` property. The scope of the element reference shall be limited by the frame reference, if one is specified, or the current frame, if none is specified.

Note: Content authors should take care to ensure that uses of these properties do not lead to unexpected navigation behaviour. In particular, it is possible that, through variations in content layout, an element which might appear on the left or right in one layout context may appear on the right or left in another layout context; e.g., due to differences in default fonts, font sizes, text layout algorithms, etc.

6.4 CSS properties introduced in this specification

6.4.1 Palettes

6.4.1.1 Introduction to the 'dde-clut' property (informative)

Authors may specify element colors using the CSS TV Profile properties indicated above. They can also specify a palette to be used on lower end receivers where color capabilities are limited. On such receivers, if a specified color is not in the palette, it is expected that the user agent will attempt to match the specified color to a color within

the palette that is considered close. If the specified color is in the palette, and the palette is of a size supportable by a receiver, then the author can expect that a User Agent will use the specified color.

6.4.1.2 The ‘dde-clut’ property

To specify that a particular palette (often referred to as a color lookup table or “clut” for short) should be used in the rendering of objects in the body of an HTML page, the author uses the ‘dde-clut’ property.

```
‘dde-clut’
      Value:              <url>| none

      Initial:            fixed values

      Applies to:         body

      Inherited:          yes

      Percentage Values:  N/A

      Media type:         dde-tv
```

The <url> value specifies the location of the actual palette. The <url> value indicates content which contains a clut in the PNG format [PNG] or in the CLT format described next.

CLT Format

The DDE-Transitional-defined clut resource is an XML file. Its syntax is defined by the DTD below.

The PublicLiteral to be used for specifying this DTD in document type declarations of the XML file is:

```
"-//SMPTE//DTD CLUT Resource 1.0"
```

and the URL for the SystemLiteral is:

```
  "http://www.smpte.org/dde-t/dtd/clutresource-1-0.dtd".

<!ELEMENT paletteResource (colorDefinition+)>
<!ATTLIST paletteResource
   colorModel   ( rgba | yuv | rgb )   #REQUIRED
   nbColors  CDATA        #REQUIRED
   firstColor   CDATA        #REQUIRED
>

<!ELEMENT colorDefinition (amtNext+)>


<!ELEMENT amtNext (CDATA)>
```

Attributes nbColors, firstColor, and amtNext are all non-negative integer values.

The value in nbColors is the number of colors in the palette.

The amtNext values are supplied in tuples, the size of which is determined by the value specified for colorModel.

If either a colorModel of rgb or yuv is specified, all of the amtNext values range between 0 and 255 as in the color property from CSS2 [CSS2]. In this case, these values refer to the amount of red, green, and blue, in that order, in the case where the colorModel is rgb; and to luminance(y) and the difference between the luminance and red (u) and between luminance and blue (v), in that order where the colorModel is specified as yuv, as defined in CCIR 601[CCIR-601].

If the colorModel is specified as rgba, then the first three values for each color correspond to the amount of red, green, and blue, in that order, as described above for the case where the colorModel is defined to be rgb. The fourth value for each color is an integer ranging between 0 and 100 (representing a percentage) and specifies the amount of opacity for that color; 100 signifying completely opaque and 0 signifying completely transparent.

The purpose of the firstColor value is to allow the simultaneous use of multiple cluts, each defining a portion of the color space. That is, the author may specify multiple concurrent cluts to be in effect so long as the portion of the color space that they occupy is non-intersecting, e.g., one clut may contain colors whose indices range between 0 and 24, inclusive, while the next clut may contain colors whose indices range between 25 and 31. The result is undefined if an author specifies multiple cluts indicating different colors for the same index.

6.4.2 Controlling the virtual keyboard display

6.4.2.1 Introduction to the virtual keyboard property (informative)

The property described in this section is used to control when, if at all, a virtual keyboard will be made available for the viewer to enter information. Sometimes it is easier for the viewer to enter numerical information by hitting the numbers directly on the remote control, rather than navigating a virtual keyboard.

6.4.2.2 Virtual keyboard property

The following CSS property is used by the author to control the appearance of the virtual keyboard.

```
'dde-virtual-keyboard'

Value: disable | enable | auto

Initial:  enable

Applies to:  all input elements

Inherited:   no

Percentage Values:  N/A

Media type:  dde-tv
```

The value "`disable`" means that the author prefers that the virtual keyboard not be presented when the viewer wants to enter data into the area, e.g., they must enter numbers via the remote control instead. The value "`auto`" means that when the element to which the property applies receives focus, the author prefers that the virtual keyboard automatically be presented to the viewer. The value "`enable`" means that the author prefers that the virtual keyboard automatically be presented to the viewer when the viewer selects the element to which the property applies. If the viewer's preferences indicate that there is an alternate preferred non-virtual keyboard available, then the alternate input device, rather than the virtual keyboard, is expected to be used even if the value has been set to enable or auto.

7 ECMAScript compliance

In addition to the restrictions of ECMA 262 (version 3) [ECMA-262] that are described in ECMA 327 (version 3) [ECMA-327], authors using ECMAScript to control interactive television on less capable receivers, have additional restrictions as follows:

- In order to avoid the necessity of searching the Scope Chain to resolve a property, support may only be provided for property identification when the object to which the property belongs is explicitly specified.
- The Arguments Object may be read only.
- The throw and try statements need not be available.
- The parseInt(), parseFloat(), and URI handling function properties of the global object may not be available.

- Regular expressions may not be supported. In particular, the RegExp() constructor property of the global object may not be available. The replace() property for Strings need not support regular expressions. (See below for additional restrictions on the replace() property.)
- Errors may not be supported. In particular, the Error(), EvalError(), RangeError(), ReferenceError(), SyntaxError(), TypeError(), and URIError() constructor properties of the global object may not be available.
- The apply() and toString() properties of function objects may not be available.
- The sort() and splice() properties of array objects may not be available.
- The match(), search(), toLocaleLowerCase(), and toLocaleUpperCase() of string objects may not be available. Also, in addition to possibly not accepting a regular expression as a parameter (i.e., for the searchValue or replaceValue parameters) in the replace() property for string objects, an implementation may not support the $replacements. When $replacements are not supported, the replace() method results in a string value derived from the original input string by replacing each matched substring with *newstring*, where *newstring* is the result of converting the replaceValue argument to a string .
- The toExponential() and toPrecision() properties of Number objects need not be available.
- None of the value properties of the Math Object need be available. Additionally, only the ceil(), floor(), random(), and round() method properties of the Math Object need be available.

8 Document and host object models

8.1 DOM and host object basis

The document object model interfaces presented below, which are modified from the respective DOM, Level 2 W3C recommendations [DOM2-Core, DOM2-HTML, DOM2-Style, DOM2-Events] will be available to authors in all interactive television environments where the hasFeature method returns true when invoked with the strings “DDE-T” and “1.0”. No new attributes or methods, different from those specified in the respective DOM, Level 2 specifications are defined. Those attributes and methods that are defined in the DOM Level 2 specifications, but may not be available for authors using a DDE-T, version 1.0 environment are noted after the specification of each adapted interface. Interfaces not defined below may not be available to a DDE-T, version 1.0 author. The definitions of the attributes and methods by the same name as those presented in the DOM Level 2 specifications are identical to the definitions presented in those specifications. Binding of the attributes and methods to ECMAScript objects is identical to that specified in the respective DOM Level 2 specifications. Since DDE-T, version 1.0 of ECMAScript need not support exceptions (also the case in DDE-1[DDE-1]), none of the methods that are designated as raising exceptions in the DOM 2 specifications will be designated as raising exceptions here.

Finally, although not absolutely necessary, for clarity, the interfaces have been re-named to add a prefix of “dde.” This prefix is only meant for clarity within this specification and any binding, present or future, of the idl interfaces to a programming language by W3C should be considered as though there were no such prefix on these interfaces.

The host object basis derives from SMPTE’s DOM-0 specification[DOM-0] as described in Section 8.1.5.

8.1.1 DOM adapted from DOM level 2 core

The following interfaces are adapted from DOM Level 2 Core[DOM2-Core]. As stated above, the types of the properties have not been modified, no new properties have been added, and definitions of the various properties and their binding to ECMAScript are precisely as defined in that specification.

8.1.1.1 Document interface

```
interface ddeDocument: ddeNode {

  readonly attribute ddeDocumentType doctype;

  readonly attribute ddeDOMImplementation implementation;

  ddeElement getElementById(in DOMString elementId);

  ddeNodeList getElementsByTagName(in DOMString tagname);
```

```
};
```

The following may not be available in the document interface: createComment(), createCDATASection(), createProcessingInstruction(), createEntityReference(), documentElement, createElement(), createDocumentFragment(), createTextNode(), createAttribute(),importNode(), createElementNS(), createAttributeNS(),and getElementByTagNameNS().

8.1.1.2 Node interface

```
interface ddeNode {

// NodeType

      const unsigned short      ELEMENT_NODE                  = 1;

      const unsigned short      ATTRIBUTE_NODE                = 2;

      const unsigned short      TEXT_NODE                     = 3;

      const unsigned short      CDATA_SECTION_NODE            = 4;

      const unsigned short      ENTITY_REFERENCE_NODE         = 5;

      const unsigned short      ENTITY_NODE                   = 6;

      const unsigned short      PROCESSING_INSTRUCTION_NODE   = 7;

      const unsigned short      COMMENT_NODE                  = 8;

      const unsigned short      DOCUMENT_NODE                 = 9;

      const unsigned short      DOCUMENT_TYPE_NODE            = 10;

      const unsigned short      DOCUMENT_FRAGMENT_NODE        = 11;

      const unsigned short      NOTATION_NODE                 = 12;

  readonly attribute unsigned short nodeType;

  readonly attribute ddeNode parentNode;

  readonly attribute ddeNodeList childNodes;

  readonly attribute ddeNode firstChild;

  readonly attribute ddeNode lastChild;

  readonly attribute ddeNode previousSibling;

  readonly attribute ddeNode nextSibling;

  boolean hasChildNodes();

};
```

The following may not be available in the node interface: nodeName, nodeValue, attributes, ownerDocument, insertBefore(), replaceChild(), removeChild(), appendChild(),

normalize(), isSupported(), namespaceURI, prefix, localName, hasAttributes(), and cloneNode().

8.1.1.3 NodeList inteface

```
interface ddeNodeList {

  ddeNode item(in unsigned long index);

  readonly attribute unsigned long length;

};
```

8.1.1.4 Element interface

```
interface ddeElement {

  readonly attribute DOMString tagName;

};
```

The following may not be available in the element interface: getAttribute(), setAttribute(), getAttributeNode(), setAttributeNode(), removeAttributeNode(), getElementByTagName(), getAttributeNS(), setAttributeNS(), removeAttributeNS(), getAttributeNodeNS(), setAttributeNodeNS(), getElementsByTagNameNS(), hasAttributes(), hasAttributeNS(), and removeAttribute().

8.1.1.5 DocumentType interface

```
interface ddeDocumentType : ddeNode{

  readonly attribute DOMString   Name;

};
```

The following may not be available in the documentType interface: entities, notations, publicId, systemId, and internalSubset.

8.1.1.6 DOMImplementation interface

```
interface ddeDOMImplementation {

  boolean hasFeature(in DOMString feature, in DOMString version);

};
```

The following may not be available in the DOMImplementation interface: createDocumentType() and createDocument().

8.1.2 DOM adapted from DOM level 2 HTML

The following interfaces are adapted from DOM Level 2 HTML[DOM2-HTML]. As stated above, the types of the properties have not been modified, no new properties have been added, definitions of the various properties and their binding to ECMAScript are precisely as defined in that specification.

8.1.2.1 HTMLElement interface

```
interface ddeHTMLElement : ddeElement{

    attribute DOMString id;

};
```

The following may not be available in the `HTMLElement interface: title, lang, dir`, and `className` attributes.

8.1.2.2 HTMLDocument interface

```
interface ddeHTMLDocument :  ddeDocument{

  attribute DOMString title;

  readonly attribute DOMString domain;

  readonly attribute DOMString URL;

  attribute ddeHTMLElement body;

  attribute DOMString cookie;

};
```

The following may not be available in the `HTMLDocument interface: referrer, images, applets, links, forms, anchors, getElementByName(), open(), close(), write()` and `writeln()`.

8.1.2.3 HTMLFormElement interface

```
interface ddeHTMLFormElement: ddeHTMLElement{

  readonly attribute long length;

  readonly attribute ddeHTMLCollection elements;

  attribute DOMString name;

  attribute DOMString action;
  attribute DOMString target;

  void submit();

  void reset();

  };
```

The following may not be available in the `HTMLFormElement interface: enctype, method`, and `acceptCharset`.

8.1.2.4 HTMLInputElement interface

```
interface ddeHTMLInputElement : ddeHTMLElement {

  readonly attribute ddeHTMLFormElement form;
```

```
  attribute boolean checked;

  attribute boolean disabled;
  attribute long maxLength;

  attribute DOMString name;

  attribute boolean readOnly;

  attribute DOMString src;

  attribute DOMString type;

  attribute DOMString value;

  void focus();

};
```

The following may not be available in the `HTMLInputElement interface`: `alt`, `blur()`, `select()`, `click()`, `useMap`, `tabindex`, `size`, `align`, `accesskey`, `accept`, `defaultChecked`, and `defaultValue`.

8.1.2.5 HTMLImageElement interface

```
interface ddeHTMLImageElement: ddeHTMLElement {

  attribute DOMString name;

  attribute DOMString src;

  attribute long height;

  attribute long width;

};
```

The following may not be available in the `HTMLImageElement interface`: `align`, `alt`, `border`, `hspace`, `longDesc`, `vspace`, `isMap`, and `useMap`.

8.1.2.6 HTMLSelectElement interface

```
interface ddeHTMLSelectElement: ddeHTMLElement{

  readonly attribute DOMString type;
  attribute long selectedIndex;

  attribute DOMString value;

  readonly attribute long length;

  readonly attribute ddeHTMLFormElement form;

  attribute boolean disabled;

  attribute DOMString name;
```

```
  void add(in ddeHTMLElement element, in ddeHTMLElement before);

  void remove (in long index);

  void focus();

};
```

The following may not be available in the `HTMLSelectElement interface`: `blur(), tabindex, size, multiple, and options`.

8.1.2.7 HTMLOptionElement interface

```
interface ddeHTMLOptionElement: ddeHTMLElement {

  readonly attribute ddeHTMLFormElement form;

  readonly attribute DOMString text;

  readonly attribute long index;

  attribute DOMString label;

  attribute boolean selected;

  attribute DOMString value;

}
```

The following may not be available in the `HTMLOptionElement interface`: `defaultSelected` and `disabled`.

8.1.2.8 HTMLTextAreaElement interface

```
interface ddeHTMLTextAreaElement: ddeHTMLElement {

  readonly attribute ddeHTMLFormElement form;

  attribute DOMString name;

  attribute boolean disabled;

  attribute boolean readOnly;

  readonly attribute DOMString type;

  attribute DOMString value;

  void focus();

  };
```

The following may not be available in the `HTMLTextAreaElement interface`: `blur(), select(), tabindex, rows, cols, accessKey`, and `defaultValue`.

8.1.2.9 HTMLAnchorElement interface

```
interface ddeHTMLAnchorElement: ddeHTMLElement{
```

```
  attribute DOMString href;

  attribute DOMString name;

  attribute DOMString target;

  void focus();

};
```

The following may not be available in the `HTMLAnchorElement interface`: `blur(), type, tabindex, shape, rev, rel, accessKey, charset, coords`, and `hreflang`.

8.1.2.10 HTMLObjectElement interface

```
interface ddeHTMLObjectElement: ddeHTMLElement{

  attribute DOMString data;

  attribute DOMString name;

  attribute long height;

  attribute long width;

}
```

The following may not be available in the `HTMLObjectElement interface: vspace, useMap, type, tabIndex, standby, hspace, form, declare, archive, border, codeBase, align, code, contentDocument`, and `codeType`.

8.1.2.11 HTMLCollection interface

```
interface ddeHTMLCollection {

  readonly attribute unsigned long length;

  ddeNode item (in unsigned long index);

  ddeNode namedItem(in DOMString name);

};
```

8.1.3 DOM adapted from DOM level 2 style

The following interface is adapted from DOM Level 2 HTML[DOM2-HTML]. As stated above, the types of the properties have not been modified, no new properties have been added, definitions of the various properties and their binding to ECMAScript are precisely as defined in that specification.

8.1.3.1 CSSProperties interface

```
interface ddeCSSProperties {

  attribute DOMString backgroundColor;

  attribute DOMString backgroundImage;

  attribute DOMString color;
```

```
  attribute DOMString fontFamily;
  attribute DOMString fontSize;
  attribute DOMString visibility;
  attribute DOMString left;
  attribute DOMString top;
  attribute DOMString borderColor;
  attribute DOMString borderWidth;
};
```

8.1.4 DOM adapted from DOM level 2 events

The following interfaces are adapted from DOM Level 2 Events[DOM2-Events]. As stated above, the types of the properties have not been modified, no new properties have been added, definitions of the various properties and their binding to ECMAScript are precisely as defined in that specification.

8.1.4.1 EventTarget interface

```
interface ddeEventTarget {
   void addEventListener (in DOMString type,
             in ddeEventListener listener,
             in boolean useCapture);
   void removeEventListener (in DOMString type,
             in ddeEventListener listener,
             in boolean useCapture);
   };
```

The following may not be available in the `EventTarget interface: dispatchEvent()`.

8.1.4.2 EventListener interface

```
interface ddeEventListener {
};
```

The following may not be available in the `EventListener interface: handleEvent()`.

8.1.4.3 Event interface

```
interface ddeEvent {
  //PhaseType
       const unsigned short CAPTURING_PHASE  = 1;
```

```
      const unsigned short AT_TARGET   = 2;

      const unsigned short BUBBLING_PHASE  = 1;

  readonly attribute DOMString type;

  readonly attribute ddeEventTarget target;

  readonly attribute ddeEventTarget currentTarget;

  readonly attribute unsigned short eventPhase;

  readonly attribute boolean bubbles;

  readonly attribute boolean cancelable;

  readonly attribute ddeDOMTimeStamp timeStamp;

  void stopPropagation();

  void preventDefault();

};
```

The following may not be available in the `EventListener interface: initEvent()`.

8.1.4.4 UIEvent interface

```
interface ddeUIEvent: ddeEvent {
readonly attribute long detail;
};
```

The following may not be available in the `UIEvent interface:view and initUIEvent()`.

8.1.4.5 HTML events

The following type of HTML events are available to the DDE-T author as defined in the DOM Level 2 Event specification: `load, unload, abort, error, select, change, submit, reset, focus, and blur.`

8.1.5 Host object model basis

The host object model basis is adopted from SMPTE's DOM-0 specification[DOM-0]. In particular, the Navigation objects, Navigator, Window, and History as specified therein comprise the basis for the host object model. The Navigator is extended with additional properties in Section 8.3.

8.2 Object model adapted from other specifications

8.2.1 Key code events

8.2.1.1 DOM3 text event extensions

This specification adapts the proposed DOM3 text event key code extensions to include many virtual key constants identified in the DASE[DASE-1] specification and several not existing in either the DOM3 working draft or in DASE to date, but found on existing remote controls. For this specification, the key codes are not bound to actual values so that content may be usable on multiple different platforms.

```
Interface ddeTextEvent: ddeUIEvent {
//Virtual Key Codes from DASE
const unsigned long VK_UNDEFINED;
const unsigned long VK_CAPS_LOCK;
const unsigned long VK_DELETE;
const unsigned long VK_DOWN;
const unsigned long VK_END;
const unsigned long VK_ENTER;
const unsigned long VK_ESCAPE;
const unsigned long VK_F1;
const unsigned long VK_F2;
const unsigned long VK_F3;
const unsigned long VK_F4;
const unsigned long VK_F5;
const unsigned long VK_F6;
const unsigned long VK_F7;
const unsigned long VK_F8;
const unsigned long VK_F9;
const unsigned long VK_F10;
const unsigned long VK_F11;
const unsigned long VK_F12;
const unsigned long VK_HOME;
const unsigned long VK_INSERT;
const unsigned long VK_LEFT;
const unsigned long VK_LOCK;
const unsigned long VK_DOWN;
const unsigned long VK_UP;
const unsigned long VK_PRINTSCREEN;
const unsigned long VK_SCROLL_LOCK;
const unsigned long VK_POWER;
const unsigned long VK_INFO;
const unsigned long VK_PINP_TOGGLE;
const unsigned long VK_TELETEXT;
const unsigned long VK_HELP;
const unsigned long VK_GUIDE;
const unsigned long VK_CHANNEL_UP;
const unsigned long VK_CHANNEL_DOWN;
const unsigned long VK_VOLUME_UP;
```

```
const unsigned long VK_VOLUME_DOWN;
const unsigned long VK_MUTE;
const unsigned long VK_STOP;
const unsigned long VK_PAUSE;
const unsigned long VK_FAST_FWD;
const unsigned long VK_RECORD;

//Modifier Codes – from DOM3 Working Draft
const unsigned long LEFT_ALT;
const unsigned long LEFT_CONTROL;
const unsigned long LEFT_META;
const unsigned long LEFT_SHIFT;
const unsigned long RIGHT_ALT
const unsigned long RIGHT_CONTROL
const unsigned long RIGHT_META
const unsigned long RIGHT_SHIFT;

//Virtual Key Codes New Here
const unsigned long RC_VK_TV;
const unsigned long RC_VK_SET_UP;
const unsigned long RC_VK_RADIO;
const unsigned long RC_VK_NAV;
const unsigned long RC_VK_MENU;
const unsigned long RC_VK_SELECT;
const unsigned long RC_VK_EXIT;
const unsigned long RC_VK_RED;
const unsigned long RC_VK_GREEN;
const unsigned long RC_VK_YELLOW;
const unsigned long RC_VK_BLUE;
const unsigned long RC_VK_RESUME;
const unsigned long RC_VK_SINGLE_STEP_FORWARD;
const unsigned long RC_VK_SINGLE_STEP_REVERSE;
const unsigned long RC_VK_FAST_REVERSE;
const unsigned long RC_VK_CUT;
```

```
const unsigned long RC_VK_COPY;
const unsigned long RC_VK_PASTE;
const unsigned long RC_VK_MIXING;
const unsigned long RC_VK_MAGNIFY;
const unsigned long RC_VK_CONTENT;
const unsigned long RC_VK_REVEAL;
const unsigned long RC_VK_VCR;
const unsigned long RC_VK_SATELLITE_DEL;
const unsigned long RC_VK_CABLE_DEL;
const unsigned long RC_VK_TERR_DEL;
const unsigned long RC_VK_DISPLAY_CLOCK;
const unsigned long RC_VK_SET_CLOCK;
const unsigned long RC_VK_COLOR_UP;
const unsigned long RC_VK_COLOR_DOWN;
const unsigned long RC_VK_BRIGHT_UP;
const unsigned long RC_VK_BRIGHT_DOWN;
const unsigned long RC_VK_CONTRAST_UP;
const unsigned long RC_VK_CONTRAST_DOWN;
const unsigned long RC_VK_PREVIOUS_CHANNEL;
const unsigned long RC_VK_PREFERENCES;
const unsigned long RC_VK_PARENTAL_CONTROL;
const unsigned long RC_VK_BOX_OFFICE;
const unsigned long RC_VK_PURCHASE;
const unsigned long RC_VK_PPV_SERVICES;
const unsigned long RC_VK_GO_ONLINE;
const unsigned long RC_VK_EXIT_APP;
const unsigned long RC_VK_SHOW_INTERACTIVE;
const unsigned long RC_VK_PINP_MOVE;

attribute DOMString outputString;
attribute unsigned long keyVal;
attribute unsigned long virtkeyVal;
attribute boolean visibleOutputGenerated;
attribute boolean numPad;
boolean checkModifier(in unsigned long modifier);
```

```
}
Unlike DASE, the modifier codes specified as symbolic constants above are only for
use in the checkModifier method and the virtKeyVal will not contain these values.
```

Attributes:

`VK_* of type unsigned long, const`

Symbolic representations of keys with specific meanings. Note that compliant content written to this specification using a given key does not guarantee that the key exists in an implementation.

`RC_VK_* of type unsigned long, const`

Symbolic representations of keys with specific DTV and remote control meanings. This specification assumes that any mapping of these symbolic representations to internal constants does not conflict with the virtual keyboard values specified in DOM L3 Events. Note that compliant content written to this specification using a given key does not guarantee that the key exists in an implementation.

`keyVal of type unsigned long`

The value of `keyVal` holds the Unicode character associated with the depressed key. If the key has no Unicode representation or no Unicode character is available the value is 0.

`numPad of type boolean`

The `numPad` attribute indicates whether or not the key event was generated on the number pad section of the keyboard. If the number pad was used to generate the key event the value is true, otherwise the value is false.

`outputString of type DOMString`

`outputString` holds the value of the output generated by the key event. This may be a single Unicode character or it may be a string. It may also be null in the case where no output was generated by the key event.

`virtKeyVal of type unsigned long`

When the key associated with a key event is not representable via a Unicode character `virtKeyVal` holds the virtual key code associated with the depressed key. If the key has a Unicode representation or no virtual code is available the value is `DOM_VK_UNDEFINED`.

`visibleOutputGenerated of type boolean`

The `visibleOutputGenerated` attribute indicates whether the key event will normally cause visible output. If the key event does not generate any visible output, such as the use of a function key or the combination of certain modifier keys used in conjunction with another key, then the value will be false. If visible output is normally generated by the key event then the value will be true. The value of `visibleOutputGenerated` does not guarantee the creation of a character. If a key event causing visible output is cancelable it may be prevented from causing visible output. This attribute is intended primarily to differentiate between keys events which may or may not produce visible output depending on the system state.

checkModifier

The `checkModifier` method is used to check the status of a single modifier key associated with a `TextEvent`. The identifier of the modifier in question is passed into the `checkModifier` function. If the modifier is triggered it will return true. If not, it will return false.

Parameters

modifier of type `unsigned long`

The modifier which the user wishes to query.

Return Value

`boolean` The status of the modifier represented as a boolean.

No Exceptions

8.2.1.2 Binding of the TextEvent Interface to ECMAScript

`Prototype Object` **`TextEvent`**

The **`TextEvent`** class has constants, of type Number, listed above from VK_UNDEFINED through RC_VK_PINP_MOVE. These symbolic constants are not bound to particular number values by this specification.

`Object TextEvent`

The `TextEvent` object has the following properties:

`outputString`

This property is of type **String**.

`keyVal`

This property is of type **Number**.

`virtKeyVal`

This property is of type **Number**.

`numPad`

This property is of type **Boolean**.

`visibleOutputGenerated`

This property is of type **Boolean**.

The **`TextEvent`** object has the following methods:

`checkModifier (modifierArg)`

This method returns a **Boolean**.

The `modifierArg` is of type **Number**.

8.3 Object model introduced in this specification

8.3.1 Dynamic cache hints

8.3.1.1 Introduction to dynamic cache hints (informative)

Since typically the amount of information that can be presented on a television screen is substantially less than contained in a page that is typically viewed on a PC, an author creating content for television will most often spread the same amount of information over multiple pages. Hence, the viewer will typically “scroll” between pages, and their navigation through a page can be a good indicator of which resources will be needed next. An author making

use of such information by conveying hints based upon this navigation to the user agent can enable much better performance on lower end clients.

8.3.1.2 Host Cache interface

The cache interface supports two methods, `prefetch` and `remove`. The `prefetch` method specifies both the URL associated with the resource to be prefetched as well as a priority indicating how likely it is that the viewer will need that resource

The cache priority value is a non-negative integer. The author can use a cache priority value of 0 to indicate that the referenced content is useful, but that the author may be unsure of its likelihood of use in comparison with other items that they are requesting to be cached. The author can use a cache priority value of 1 to indicate the belief that caching the specified resource is very important; in fact, only other requests also with a cache priority value of 1 may be of the same or more importance. A very large value for the priority indicates that a resource will likely not be used (hence informing the user agent that it may reclaim the memory currently used to hold the URL's associated resource in cases where it is needed).

The `remove` method may be used to remove a cached copy of the resource associated with the URL argument. Since it is to be removed from the cache, and not just invalidated, the system will not waste resources re-validating the entry. Note that invoking the remove method is different from assigning a very large integer as the cache priority value in that assigning such a large integer value only makes the space used to store that resource more available for garbage collecting and/or to hold high priority resources.

```
Interface cache {
  void prefetch(in DOMString URL, in short priority);
  void remove (in DOMString URL);
};
```

8.3.1.3 Binding of the cache interface to ECMAScript

The `Cache Object`, which implements the cache interface above, is accessible as a property of the `Navigator (Navigator::cache).`

`Object` **`cache`**

The **`cache obect`** has the following methods:

`prefetch``( URLArg, priorityArg)`
`This method does not return a value.`

The `URLArg` is of type **`String`**.

The `priorityArg` is of type **`Number`**.

`Remove``(URLArg)`
`This method does not return a value.`

The `URLArg` is of type **`String`**.

9 Media types

The following media types are available to the author: MPEG-2 video[MPEG2-Video], jpeg[JPEG], png[PNG], and MPEG-2 audio[MPEG2-Audio].

10 Locators

10.1 The http: and https: protocol locators

Any content authored according to this specification may use, where a URI [URI] is allowed, any URI that is formulated according to the HTTP 1.1 specification [HTTP-1.1]. The https:// locator [HTTPS] is also useful for authors designing content for use where an interaction channel is available.

10.2 The tv: protocol locator

Any content authored according to this specification may use, where a URI [URI] is allowed, any URI that is formulated according to the tv: specification [URI-TV].

10.2.1 Query syntax useful with the tv: protocol locator

Similar to the query syntax documented in section 5.1.2.3.1.4 of the DASE, Part 1 [DASE-1], a URI which employs this scheme may specify a query component. The form of a query component shall adhere to the following syntax which is only slightly different from that specified by DASE:

```
query            : query_component [ “;” query_component  ]   *
query_component     : component { “=” component_name ]
component           : “video” | “audio”
component_name         : pchar+
pchar            : { any printable character except ‘:’  }
```

A query that specifies a `video` component resolves to either a video elementary stream or an aggregate of program elements that includes a video elementary stream.

A query that specifies an `audio` component resolves to either an audio elementary stream or an aggregate of program elements that includes an audio elementary stream.

When specified, a component name is used to resolve to a part or whole of a particular program element that is labeled with the component name.

Annex A (informative)
Bibliography

In addition to the normative references listed in Section 2, he following references were used, in part, in formulating this specification.

Abbreviation	Reference
[BML]	STD-B24v1.2 Vol. 2, ARIB: Digital Coding and Transmission Specification for Digital Broadcasting, XML-based Multimedia Coding Scheme (June 20th, 2000).
[CSS2]	REC-CSS2-19980512, W3C Recommendation Cascading Style Sheets, level 2 (CSS2), (May 12, 1998).
[CSS2R1]	WD-CSS21-20020802, W3C Working Draft: Cascading Style Sheets, level 2, Revision 1 (CSS2.1 Specification), August 2nd, 2002.
[CSS3-Box]	WD-css3-box-20021024, W3C Working Draft: CSS3 module: The Box Model (October 24th, 2002).
[CSS3-UI]	WD-css3-ui-20020802, W3C Working Draft: CSS3 module: Basic User Interface (August 2nd, 2002).
[DASE-1]	DASE-1-PS-20021105-PDF, ATSC Proposed Standard: DTV Application Software Environment (DASE), November 5, 2002..
[DDE-1]	SMPTE 363M, Declarative Data Essence – Content Level 1 (2002).
[DOM3-EVENT]	WD-DOM-Level-3-Events-20020712, W3C Working Draft: Document Object Model (DOM) Level 3 Events Specification Version 1.0 (July 12th, 2002).
[DVB-HTML]	TS 102 812 V1.1.1, ETSI: The DVB Multimedia Home Platform (DVB-MHP) 1.1, (November, 2001).
[MIME]	RFC 2045, IETF: Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies (November 1997).
[OCAP-1.0]	OC-SP-OCAP1.0-I04-021028, CableLabs: OpenCable Application Platform Specification (OCAP) 1.0 (October 28, 2002).
[Porter-Duff]	"Compositing Digital Images", T. Porter, T. Duff, SIGGRAPH '84 Conference Proceedings, Association for Computing Machinery, Volume 18, Number 3, July 1984.
[URI]	RFC2396, IETF: Uniform Resource Identifiers (URI): Generic Syntax (August 1998).
[XHTML1.0]	REC-xhtml1-20020801, W3C Recommendation: XHTML™ 1.0 – The Extensible HyperText Markup Language, 2nd edition (August 1st, 2002).