EG 39-2003

SMPTE ENGINEERING GUIDELINE

Overview of Declarative Data Essence

Table of contents

1 Scope

This guideline provides an overview of the declarative data essence standards, describes how the various documents and technical components are related, and provides informative material useful to the users of these standards.

2 Introduction

The initial group of standards were developed in SMPTE based on the Advanced Television Enhancement Forum (ATVEF) specification [ATVEF]. These are collectively known as declarative data essence (DDE) derived from the terminology developed in the joint SMPTE/EBU work found in [SMPTE-EBU]. This was further labeled as content level 1 after the ATVEF specification for 1.0, and in anticipation of both lower and higher content levels. Hence, the shorthand, DDE-1.

The ATVEF specification was broken into six separate SMPTE documents that cover the original ATVEF specification. These specifications are:

SMPTE 343M-2002, Television — Declarative Data Essence — Local Identifier (lid:) URI Scheme

SMPTE 357M-2002, Television — Declarative Data Essence — IP Multicast Encapsulation

SMPTE 361M-2002, Television — NTSC IP and Trigger Binding to VBI

SMPTE 363M-2002, Television — Declarative Data Essence — Content Level 1

SMPTE 364M-2001, Television — Declarative Data Essence — Unidirectional Hypertext Transport Protocol

Approved January 8, 2003

SMPTE 366M-2002, Television — Document Object Model Level 0 (DOM-0) and Related Object Environment

The relationship of all six of these documents can be found in figure 1.

Finally, it is worth noting that there is ongoing work (not covered in this guideline) on wrappers for this content for carriage in SMPTE KLV; and there is very early work on a content level 2.

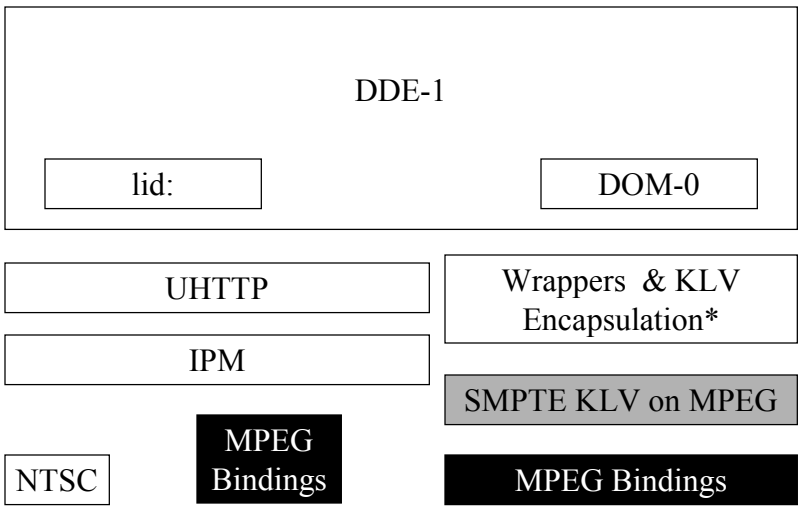

Figure 1 – Relationship of the DDE-1 standards

The collection of DDE-1 documents is fully ATVEF compliant, and is collectively known as Content Level 1, or DDE-1. No extensions were designed, and no new functionality was added. However, a considerable amount of new material was added to more fully specify the work for the purpose of providing interoperability. Specifically, there was significant additional work put into the following technology:

- DOM-0
- Triggers (defined in [DDE-1])
- Lid:

The DDE-1 document set is an authoring content standard. As such, it avoided as much as possible specific receiver behavior. However, expectations of receiver behavior are implied, and often overtly stated. Some more information on the expected behavior is covered in this guideline.

3 Authoring guidelines

3.1 HTML4

Some operations taken for granted in computer-based browsers that decode HTML should often be avoided when using DDE-1. In general, anything that would require extensive use of an input device, such as pull-downs and similar input objects. In addition, the expectation of scrolling down or to the right to view a page may often result in parts of the display not being viewable due to limited (or unwilling) user input. It is a good idea to create pages that are a single screen, and avoid any assumption of input devices, certainly anything more than can be done on a standard remote control.

Absolute lengths and sizes should be avoided to permit consistent rendering on displays of varying resolutions. Using pixel coordinates will result in significant undesirable variations.

DDE-1 does not provide good means to ensure accurate positioning of the HTML elements with respect to the video. Since the video format can be altered during distribution from its original format when the DDE-1 was created, this poses interesting registration challenges at the decoder. For example, it can be as gross a change as a conversion from 16x9 to 4x3 letterbox. Authors are cautioned about making any assumptions about the registration between the video and the DDE-1 content at the decoder when the distribution is not well known. Authors should try to stay within the safe area and safe titling areas as defined in SMPTE RP26.3 [SAFE].

Authors are referred to the general interoperability warnings found within the HTML specification [HTML].

3.2 Style sheets and fonts

There is no default style sheet in DDE-1, and there is no requirement that a receiver choose any particular set of styles. Thus, without authoring and sending a style sheet along with the HTML content, the variation in rendering across manufacturers will be significant and perhaps undesirable. Authors are encouraged to make explicit use of CSS style sheets in order to control the display of their content.

As in HTML for positioning, percent sizes should be used wherever possible to provide display independence.

DDE-1 does not support downloadable fonts; however, there are two default fonts with specific font name sizes required to be supported. Authors are encouraged to make use of these fonts for better display control. The font sizes should be specified by their font name sizes.

3.3 ECMAScript

Authors should avoid using eval() that results in dynamically generated HTML. Doing this makes transcoding of the content computationally impossible, and thus may affect the quality in future generation enhancement systems.

3.4 DOM-0

Authors should avoid using document.write() with arguments other than constants. Doing this makes transcoding of the content computationally impossible, and thus may affect the quality in future generation systems.

3.5 URI schemes

DDE-1 supports the URI schemes tv: [TV], lid: [LID], javascript: [DDE-1] and http: [HTTP]. Only http: may be used for DDE-1 content that must be fetched from the Internet, which is always the case on Transport-A-only systems. The lid: scheme may not be used for Transport-A-only systems, because lid: implies that the content is broadcast as DDE-1 and cached locally, which is not the case with Transport A. On Transport-B systems, both lid: and http: schemes may be used as follows: either lid: or http: can be used for DDE-1 content that is broadcast and cached locally; only the http: scheme should be used for content that is not broadcast and not available locally, because the lid: scheme only works for locally cached content. Authors are specifically cautioned against using the common schemes, ftp:, https:, and shttp:.

NOTE – The javascript: URI scheme is supported with certain limitations (see annex A for details).

3.6 UUID

UUIDs would normally be generated automatically by the authoring tool through a call to the operating system to obtain one. UUIDs are often also called guids, and are available in some form on all popular operating systems.

UUIDs are constructed in part from the MAC address of the network adapter being used in the computer doing the authoring. In the rare occurrence that a UUID needs to be generated from a device without a network interface, then care must be taken to use an unused MAC address. This can be done by setting the most significant bit of the MAC address field (MAC address assignments use this to signal multicast destinations and would never be used for a real address). Alternatively, a statistically unique value can be generated through the hash of the field(s) as described further in RFC 2518, section 6.4.1 [WEBDAV].

4 Receiver behavior

This clause provides information which should be considered when implementing a decoder that decodes the DDE-1 content. Since DDE-1 content is an authoring standard and not a decoder standard, this may be helpful to manufacturers of any decoding equipment, including consumer electronics receivers. The term *receiver* here is generally meant to refer to any type of decoder.

4.1 Receiver model

Figure 2 shows the receiver block diagram.

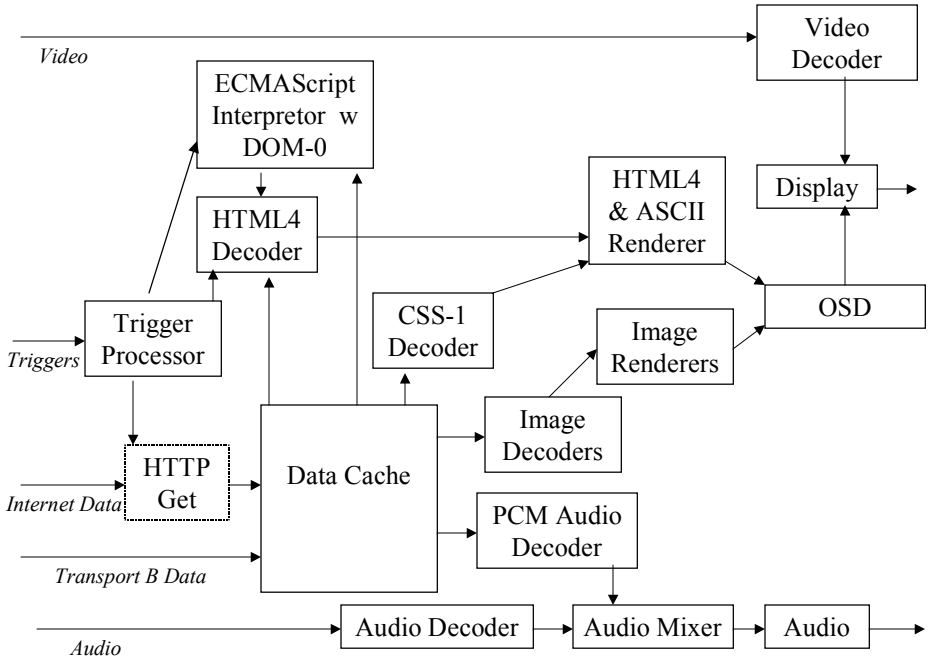

Figure 2 – Receiver block diagram

4.2 Enhancement characterization

An enhancement is defined as content added to a video/audio service. An enhancement can further be described by its behavioral characteristics as specified in [DDE-1]. From this point of view, an enhancement is a sequence of topmost HTML documents whose content is specified in [HTML]. The first HTML document of an enhancement, the initial topmost document, is always instantiated by means of a trigger. Subsequent topmost documents within an enhancement are instantiated as a result of a navigation from the current document initiated either by a trigger or a viewer selection (see figure 3).

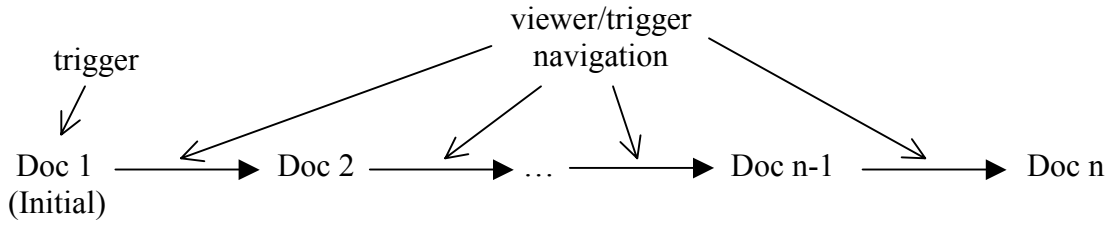

Figure 3 – DDE-1 enhancement

4.3 State model

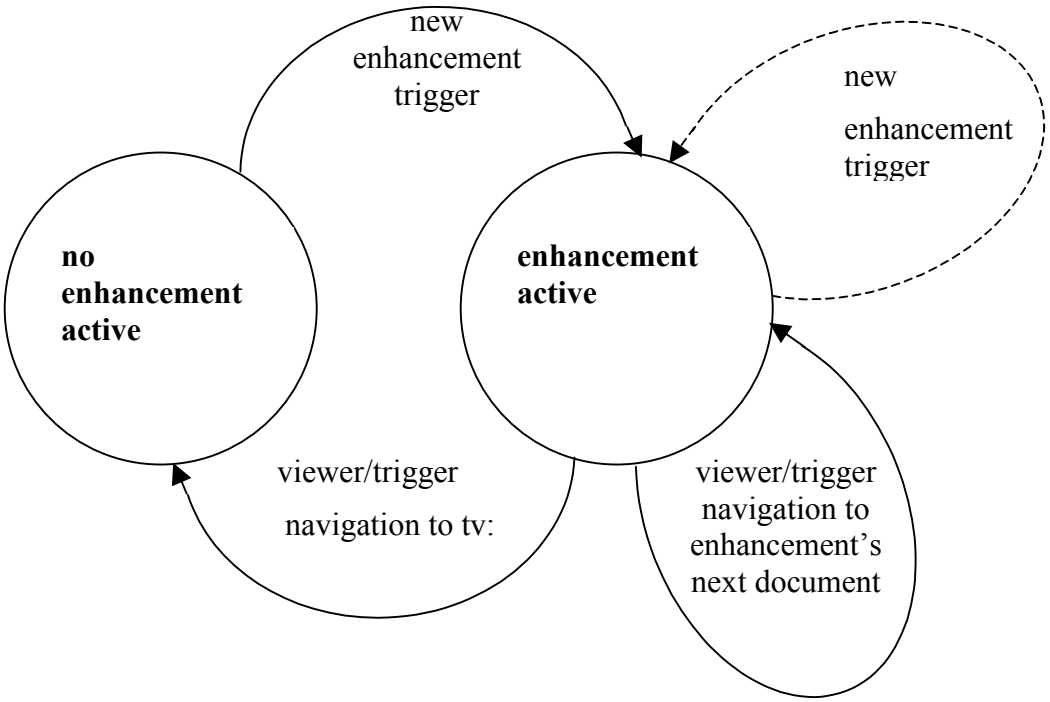

Figure 4 – Diagram for enhancement behavior

Figure 4 shows the state diagram for enhancement behavior. Table 1 describes the basic types of triggers. The enhancement state model is described from the point of view of the state transition arcs in figure 4. Each state transition can result from either a new enhancement trigger or a trigger for the current enhancement.

Table 1 – Basic trigger types

Trigger type	Enhancement active?	Trigger Description[1]
New enhancement trigger	NO	Trigger with "name" attribute, and URL DIFFERENT from the last topmost document of immediately preceding enhancement
	YES	Trigger with "name" attribute, and URL DIFFERENT from current enhancement's current topmost document[2]
Trigger for current enhancement	YES	Trigger with URL EQUAL the URL of the current enhancement's current topmost document[2]

NOTES

[1] When determining whether two URLs are DIFFERENT or EQUAL, characters in the URLs including and following the first ? or # are ignored in the comparison.

[2] May result in implementation specific behavior.

4.3.1 New enhancement triggers

As shown in table 1, a new enhancement trigger is a trigger with a name attribute, and with either:

- a URL different from the last topmost document of the immediately preceding enhancement if no enhancement is active; or
- a URL different from the current enhancement's current topmost document if an enhancement is active.

New enhancement triggers are used to initiate a new enhancement.

From the no enhancement active state, an enhancement is initiated by a new enhancement trigger. An enhancement can only be initiated by a new enhancement trigger.

From the enhancement active state, a new enhancement trigger may replace the current enhancement with a new enhancement in an implementation specific manner. In order to ensure interoperable behavior, before sending a new enhancement trigger, an enhancement should be terminated by means of a viewer/trigger navigation which replaces the enhancement's current topmost document with tv:.

4.3.2 Triggers for the current enhancement

As shown in table 1, a trigger for the current enhancement is a trigger with a URL equal to the URL of the current enhancement's current topmost document. Triggers for the current enhancement are used to:

- navigate to an enhancement's next topmost document;
- terminate an enhancement;
- change the display and/or state of an enhancement.

In an enhancement active state, a viewer/trigger navigation to enhancement's next document replaces the current topmost document of the enhancement with the contents of the enhancement's next document. With viewer initiated navigation, the current topmost document is replaced as the result of a viewer navigation (e.g., by means of a link) to another HTML document. With trigger initiated navigation, the current topmost document of the enhancement is replaced as the result of a trigger script (such as: window.top.location.href='<URL for enhancement's next document>';) being executed. Such a trigger script is executed upon receipt of a trigger for the current enhancement. In order for the enhancement to receiver the trigger, the enhancement's current topmost document must contain a trigger receiver object.

NOTE – DDE-1 only requires the capability for running one enhancement at a time.

Even though a new topmost document is being displayed, that document is still part of the current enhancement. However, note that triggers for the current enhancement delivered subsequent to the activation of the new document must have a URL equal to the URL of the new document. If this is not the case, then either the trigger is ignored (if it does not have a name attribute), or may initiate a new enhancement in an implementation specific manner (if the trigger does have a name attribute). In order to ensure interoperable behavior, before sending a new enhancement trigger, an enhancement should be terminated by means of a viewer/trigger navigation which replaces the enhancement's current topmost document with tv:.

Note that triggers for the current enhancement may have a name attribute. A name attribute in a trigger for the current enhancement does not initiate a new enhancement. Including a name attribute in a trigger can be used to achieve the following effect. If a viewer tunes to a program after the initial trigger for the program's enhancement was sent, then, by including a name attribute in subsequent triggers for that enhancement, the enhancement is initiated as a result of the receipt of a subsequent trigger.

In the enhancement active state, an enhancement can be terminated by viewer/trigger navigation to tv:. More specifically, the navigation must replace the current topmost document with tv:. With viewer initiated navigation to tv:, the current topmost document is replaced with tv: as the result of a viewer navigation (e.g., by means of a link) to the URL tv:. With trigger initiated navigation, the current topmost document of the enhancement is replaced with tv: as the result of a trigger script (such as: window.top.location.href='tv:';) being executed. Such a trigger script is executed upon receipt of a trigger for the current enhancement. In order for the enhancement to receiver the trigger, the enhancement's current topmost document must contain a trigger receiver object.

An active enhancement may also terminate as a result of the receipt of a trigger to initiate a new enhancement. Please note that it is implementation specific whether, and in what manner, an

enhancement terminates upon the arrival of a trigger for a new enhancement. In order to ensure interoperable behavior, before sending a new enhancement trigger, an enhancement should be terminated by means of a viewer/trigger navigation which replaces the enhancement's current topmost document with tv:.

The display and/or state of an active enhancement can be changed as a result of the receipt of a trigger for the current enhancement. The enhancement's display and/or state is changed as a result of the execution of the trigger's script.

4.4 Cache management

4.4.1 Size

Receivers are expected to provide buffering for one megabyte (1 MB) of cached simultaneous content. Content creators who want to reach the maximum number of receivers should manage their content such that the instantaneous high-water mark of simultaneous cached content is no more than 1 MB.

All cache size values are the uncompressed sizes of the content stripped of all transport headers. Receivers may of course choose to store content compressed to reduce local memory utilization. When carried in IP Multicast and announced with SDP, the specific cache size required for each enhancement is required be specified in the announcement. The value of the SDP announcement extension, tve-size, represents the maximum size cache needed to hold content for the current page at any time during the program including all pages reachable by local links. It is the high water mark during the program, not the total content delivered during the program.

4.4.2 Behavior

The cache is generally be expected to operate according to RFC 2616 [HTTP], sections 13 and 14. This provides the semantics for handling the required, as well as optional, HTTP header fields defined in UHTTP that can affect the cache behavior, such as expires.

Expired content is not used or displayed, even if it is present in the cache when it was needed. The cache is flushed on receiver startup, but not on other conditions, including channel change, or even receipt of a new enhancement.

Content that is delivered as a single entity is not entered into the cache until all sub-components are received. That is, when using IP multicast and UHTTP for delivery, and a multipart entity is received, the individual content items would only be made available to the executing application if the entire multipart is successfully received and decoded. Partial cache updates may result in undesired composite page displays (i.e., today's news headlines with yesterday's photos).

4.5 Cookies

The general syntax for cookies is described in RFC 2965 [HTTP-STATE], section 4.2.2, which consist of name/value pairs. The names supported in DDE-1 are name and expires, with the latter being a date syntax defined as Wdy, DD-Mmm-YY HH:MM:SS GMT.

On powerup, a receiver should set the string to null.

The receiver buffer for cookies is 1024 bytes for session cookies.

The cookie string supported by DOM-0 document object concatenates the two name/value pairs of multiple cookies into a single string.

On receipt of a UHTTP containing HTTP cookie fields, the name and expires name/values are appended to the existing value of the cookie string.

When the document.cookie property is read through DOM-0, it returns the entire string.

When the document.cookie property is written to, its previous value is replaced (i.e., it is not appended). The new string needs to conform to the constraints above — only the two name/value pairs.

4.6 CSS-1 and fonts

A receiver is effectively required to support the fonts and sizes as defined in DDE-1. Note that the mapping of the named size values (i.e., xx-small, etc.) should be done such that the size constraints are valid on the given display resolution of the specific receiver.

5 Distribution issues

There are various considerations when encoding DDE-1 content into the distribution. When using a specific encapsulation, such as IP multicast, there are pros and cons to encoding it early in the distribution. The pros are that during the period where the infrastructure is not established to carry data as a peer, using IP multicast permits out of band delivery of the data stream to the final emission facility. However, this also means that decisions about bandwidth utilization, repetition rate of content items, and other such decisions have to be made upstream of the emission, and thus perhaps without knowledge of bandwidth considerations.

There is work in process to define a standard for the carriage of data content, including enough metadata to properly encode it. Readers are encouraged to keep in touch with this activity.

While challenging at this writing, authors are encouraged to defer encoding of the content to as close as possible to final emission.

6 Ad Insertion scenario details

TV programs consist of entertainment programs interspersed with commercials. Program and commercial segments are usually produced independently. There is the obvious requirement that ITV content for the different segments should not interfere with each other. There are two approaches to accomplishing this.

As shown in figure 6, the first approach is to initiate an enhancement at the beginning of a segment, and then, terminate that enhancement at the end of the segment. As described in the section on the enhancement behavior state model, a new enhancement trigger initiates a new enhancement, and navigation from the current enhancement to tv: by means of a viewer selection or a trigger, terminates an enhancement.

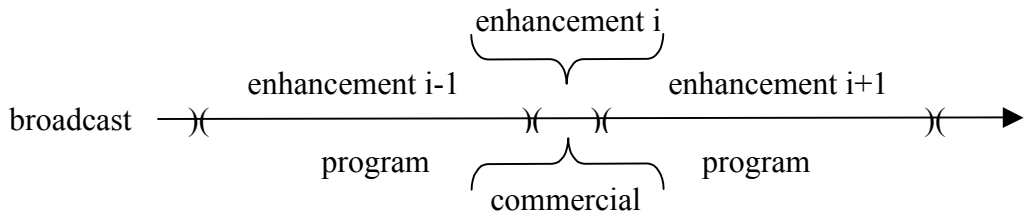

Figure 5 – Sequential enhancements

As shown in figure 6, the second approach is for several sequential segments to share a single enhancement which serves as an executive for displays associated with each segment. For example, the executive could consist of a single main frame and the displays associated with each segment are sub-frames. Frames for the segments are displayed and removed by means of triggers at the beginning and end of each segment respectively. The initiation and termination of the shared enhancement is accomplished in the same manner as the first approach.

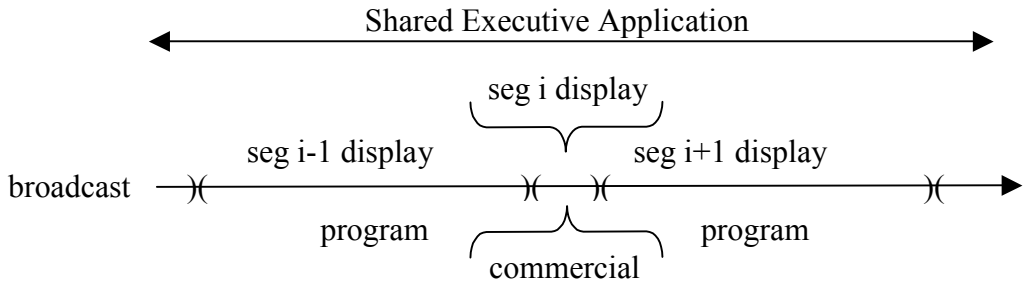

Figure 6 – Enhancement with shared executive

In both approaches, saving enhancement state between program segments may be an important design consideration. A single television program has several program segments. It may be necessary for interactive content for the entire program to appear as a single, integrated, seamless, interactive experience. This means that state information for the interactive experience must be maintained between program segments.

In the shared executive approach, maintaining state information between program segments may be accomplished by ECMAScript variables. In the sequential enhancement approach, there are basically three ways to maintain state:

- In triggers: A trigger's script attribute and the anchor and search fields of a trigger's URL can contain state information. In this case, the state information is almost always known when the content is initially produced. Putting state information in triggers is particularly useful for setting (or resetting) an enhancement to a known state.

- In a cookie: The cookie property of the document object is commonly used in Web content to maintain state between document instantiations. The technique is the same with enhancement documents.

- On a server: Maintaining state on a server is a Web technique which can apply to enhancement documents. This approach requires a return channel.

These techniques for maintaining state can also be used in combination. One example is to maintain state on a server, and return state values to the receiver in triggers. No state information need be kept on the receiver. However, this approach requires a close real-time coordination between the servers and the broadcast equipment.

7 Examples

Three enhancement examples are presented. Each example implements the same application, namely, counting arriving triggers. Each example displays a count of the triggers that are sent in the broadcast stream to the enhancement. Such triggers take the form:

<http://*pathname*>[v:1][n:count?][s:count_triggers()][*checksum*] for Transport A, or

<lid://*pathname*>[v:1][n:count?][s:count_triggers()] for Transport B.

Each example can run in a limited manner with a Web browser. Each has a link click here to show trigger arrival effect enabling the viewer to see the same effect on the display as though a trigger had arrived. Each also has a link exit enhancement to enable the viewer to terminate the enhancement. Selection of this link replaces the topmost document of the enhancement with tv:.

In order to minimize the size of each example, the complexity of the display is kept to a minimum. The goal is to illustrate the structure of each example to clarify the technique. For esthetic and performance reasons, TV enhancements may differ from normal Web pages.

7.1 No-frames

The no-frames example illustrates an enhancement consisting of a sequence of topmost documents. The display presented to the viewer changes as a result of reloading a single topmost document.

When the trigger count is incremented, the new value is added to the URL for the enhancement's document as a search string, and then the topmost document is reloaded. The search string in the URL becomes the new trigger count value displayed. This illustrates the use of a URL's search string as means of maintaining state between the sequential instantiations of an enhancement's documents. Note that this example does not maintain the trigger count value between instantiations of the no-frames enhancement.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
      "http://www.w3.org/TR/REC-html40/transitional.dtd">

<html>
<head><title>DDE-1 No-Frames Example</title>

<object type="application/tve-trigger" id="triggerReceiverObj" >
</object>

<script type="text/ecmascript">

function count_triggers() {
   ntrigs = ntrigs + 1;
   qmark = window.top.location.href.indexOf("?");
   if (qmark >= 0)
     {window.top.location.href = window.top.location.href.substring(0, qmark) +
                                 "?" + ntrigs.toString();}
      else
     {window.top.location.href = window.top.location.href + "?" +
                                  ntrigs.toString();};
}

if (window.location.search.substr(1) == "") {ntrigs = new Number(0);}
     else
   {ntrigs = new Number(window.location.search.substr(1));};

</script>

</head>
```

```
<body bgcolor="black" >

<table width="100%">

<tr>
<td align="center"><font color="blue">
<u><a onclick="count_triggers(ntrigs);">
        click here to show trigger arrival effect</a></u>
</font>
</td>
</tr>

<tr bgcolor="white">
<td><font size=medium>
<script type="text/ecmascript">
document.write("<center><h2>" + ntrigs + " triggers received<\/h2><\/center>");
</script>
</font></td></tr>

<tr>
<td align="center">
        <table>
        <tr><td>
    <img src="tv:" width=320 height=240 alt="TV Picture"><br>
        </td></tr>
        <tr>
    <td align="center"> <a href="tv:" TARGET="_self">Exit Enhancement</a>
     </td></tr>
        </table>
</td>
</tr>
</table>

</body>
</html>
```

7.2 Frameset

The frameset example illustrates an enhancement which consists of only a single topmost document which remains in place for the life of the enhancement. The display is changed by reloading frames beneath the topmost document. This example consists of six HTML documents.

The frameset example also provides the viewer with a full-screen link to enable the viewer to watch the video/audio full screen. Unlike the exit enhancement link which terminates the enhancement, selection of the full screen link navigates to a frame which maintains the trigger counting functionality in the background. In the full screen mode, the icon “=>I” is displayed over the full screen video/audio to enable the viewer to return to the interactive mode.

The no-frames example illustrates the use of a URL’s search string to maintain state between the enhancement’s document instances. The frameset example illustrates the use of a document’s cookie property to maintain state between enhancement instantiations. Sequential enhancement instantiations can be used to provide the viewer with a seamless, integrated interactive experience throughout a program when there are multiple segments of the program interspersed with commercial segments. The trigger count value and the full screen/interactive mode are saved in the document’s cookie.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
       "http://www.w3.org/TR/REC-html40/frameset.dtd">

<html>
<head><title>DDE-1 Frameset Example</title>

<object type="application/tve-trigger" id="triggerReceiverObj" >
</object >

<script type="text/ecmascript">

function count_triggers() {
    ntrigs = ntrigs + 1;
    if (window.mainframe.length > 1) {
            window.mainframe.frame2.location.href = "frame2.htm";
    };
}

function get_var_in_cookie(var_name)
{
   var v, pos, start, end;
   v = var_name + "=";
   pos = document.cookie.indexOf(v);
   if (pos != -1) {
            start = pos + v.length;  //start of save_state value
            end = document.cookie.indexOf(";", start);  // end of save_state value
            if (end == -1) end = document.cookie.length;
            v = document.cookie.substring(start, end);
            return unescape(v);
   } else return "undefined";
}

function save_cookies()
{
var ExpireDate = new Date();
var expiredays = 7;

ExpireDate.setTime(ExpireDate.getTime() + (expiredays * 24 * 3600 * 1000));

document.cookie = "ntrigs=" + ntrigs + "; expires=" + ExpireDate.toGMTString();
document.cookie = "itv=" + window.mainframe.location.href + "; expires="
                                    + ExpireDate.toGMTString();

}
//         load up variables

temp = get_var_in_cookie("ntrigs");
         if(temp == "undefined") {ntrigs = 0;} else {ntrigs = new Number(temp);};
itv = get_var_in_cookie("itv"); if(itv == "undefined") {itv = "main-itvframe.htm";};

</script>
</head>

<frameset onLoad="window.mainframe.location.href=itv" onUnload="save_cookies()">
```

```
<frame name=mainframe >
</frameset>

</html>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
      "http://www.w3.org/TR/REC-html40/frameset.dtd">

<html>
<head><title>main interactive tv frame of frames example</title></head>
<frameset rows="10%,10%,75%">

<frame name=frame1 src="frame1.htm" >
<frame name=frame2 src="frame2.htm" >
<frame name=tvframe src="tvframe.htm" >

</frameset>

</html>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
      "http://www.w3.org/TR/REC-html40/transitional.dtd">

<html>
<head><title>frame 1 of frames example</title></head>
<body bgcolor="black" >
<center><font color="blue">
<u><a onclick="top.count_triggers()">
          click here to show trigger arrival effect</a></u>
</font></center>
</body>
</html>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
      "http://www.w3.org/TR/REC-html40/transitional.dtd">
<html>
<head><title>frame 2 of frames example</title></head>
<body>
<center>
<script type="text/ecmascript">
document.write("<center><h2>" + top.ntrigs + " triggers received<\/h2><\/center>");
</script>
</center>
</body>
</html>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
      "http://www.w3.org/TR/REC-html40/transitional.dtd">

<html>
<head><title>tv frame of frames example</title></head>
<body bgcolor="black" >
<center>
          <table>
          <tr><td>
```

```
    <img src="tv:" alt="TV Picture" width=320 height=240 ><br>
        </td></tr>
        <tr><td>
    <a href="main-tvframe.htm" TARGET="mainframe">Full Screen</a>
       -----------
    <a href="tv:" TARGET="_top">Exit Enhancement</a>
     </td></tr>
        </table>
</center>
</body>
</html>



<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
      "http://www.w3.org/TR/REC-html40/transitional.dtd">

<HTML>
<head><title>main tv frame of frames example</title></head>
<BODY
 style="background:url(tv:);font-size:150%;color:blue;vertical-align:top;text-align:right">

  <a href="main-itvframe.htm" target="mainframe">=>I</a>

</BODY>
</HTML>
```

7.3 Single document frameset using document.write()

The single document frameset example using document.write() has all of the features of the frameset example. In addition, the single document frameset example using document.write() also uses the same frameset structure as the frameset example. However, the single document example using document.write() is contained within a single HTML document, whereas, the frameset example is made up of six separate HTML documents. This is accomplished by the use of document.write().

For convenience or performance considerations, it may be desirable to have all content for an enhancement contained in one document. This is a common practice in content development on the Web where display elements are contained in <DIV> sections which are then mutated/shown/hidden to change the display. This practice is not available in [DOM-0]. The single document example shows how to change the display using frames while maintaining frame contents within a single document.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
      "http://www.w3.org/TR/REC-html40/frameset.dtd">

<html>
<head><title>DDE-1 Single Document Frameset Example using 'document.write()'</title>

<object type="application/tve-trigger" id="triggerReceiverObj" >
</object>

<script type="text/ecmascript">

doctype_hdr_f = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Frameset//EN'" +
```

```
            "'http://www.w3.org/TR/REC-html40/frameset.dtd'>";

doctype_hdr_t = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'" +
            "'http://www.w3.org/TR/REC-html40/transitional.dtd'>";

main_itvframe_doc = doctype_hdr_f;
main_itvframe_doc = main_itvframe_doc +
              "<html><head><title>main_itvframe_doc<\/title><\/head>";
main_itvframe_doc = main_itvframe_doc +
            "<frameset rows='10%,10%,75%' onload='top.load_mainitvframes()'>";
main_itvframe_doc = main_itvframe_doc + "<frame name=frame1 >";
main_itvframe_doc = main_itvframe_doc + "<frame name=frame2 >";
main_itvframe_doc = main_itvframe_doc + "<frame name=tvframe >";
main_itvframe_doc = main_itvframe_doc + "<\/frameset> <\/html>";

main_tvframe_doc = doctype_hdr_t;
main_tvframe_doc = main_tvframe_doc +
                  "<html><head><title>main_tvframe_doc<\/title><\/head>";
main_tvframe_doc = main_tvframe_doc +
                  "<BODY ";
main_tvframe_doc = main_tvframe_doc +
 "style='background:url(tv:);font-size:150%;color:blue;vertical-align:top;text-align:right'>";
main_tvframe_doc = main_tvframe_doc +
           "<u><a onclick='top.load_mainframe(\"main_itvframe_doc\")'>";
main_tvframe_doc = main_tvframe_doc + "=>I<\/a><\/u>";
main_tvframe_doc = main_tvframe_doc + "<\/BODY><\/HTML>";

frame1_doc = doctype_hdr_t;
frame1_doc = frame1_doc +
          "<html><head><title>frame1_doc<\/title><\/head><body bgcolor='black' >";
frame1_doc = frame1_doc +
        "<center><font color='blue'><u><a onclick='top.count_triggers()'>";
frame1_doc = frame1_doc + "click here to show trigger arrival effect<\/a>";
frame1_doc = frame1_doc + "<\/u><\/font><\/center><\/body><\/html>";

frame2_doc = doctype_hdr_t;
frame2_doc = frame2_doc +
          "<html><head><title>frame2_doc<\/title><\/head><body><center>";
frame2_doc = frame2_doc + "<script type='text/ecmascript'>";
//   Note: the "/" in the end tags in the following string must be
//         doubly escaped in order for the document to be valid HTML 4.0
//         when it is loaded into the frame
frame2_doc = frame2_doc + "document.write('<center><h2>' + top.ntrigs + ' " +
                    "triggers received<\\\/h2><\\\/center>');";
frame2_doc = frame2_doc + "<\/script>";
frame2_doc = frame2_doc + "<\/center><\/body><\/html>";

tvframe_doc = doctype_hdr_t;
tvframe_doc = tvframe_doc +
     "<html><head><title>tvframe_doc<\/title><\/head><body bgcolor='black'><center>";
tvframe_doc = tvframe_doc +
  "<table><tr><td><img src='tv:' alt='TV Picture' width=320 height=240 ><br><\/td><\/tr>";
tvframe_doc = tvframe_doc + "<tr><td><font color='blue'><u>";
tvframe_doc = tvframe_doc +
         "<a onclick='top.load_mainframe(\"main_tvframe_doc\")' >Full Screen<\/a>";
tvframe_doc = tvframe_doc + "<\/u><\/font>----------";
```

```
tvframe_doc = tvframe_doc +
            "<a href='tv:' TARGET='_top'>Exit Enhancement<\/a><\/td><\/tr>";
tvframe_doc = tvframe_doc + "<\/table><\/center><\/body><\/html>";


function load_mainframe(itv) {
   if (itv == "main_itvframe_doc") {
      window.mainframe.document.write(main_itvframe_doc);
      window.mainframe.document.close();
   } else {
      window.mainframe.document.write(main_tvframe_doc);
      window.mainframe.document.close();
   };
}

function load_mainitvframes() {
   window.mainframe.frame1.document.write(frame1_doc);
   window.mainframe.frame1.document.close();
   window.mainframe.frame2.document.write(frame2_doc);
   window.mainframe.frame2.document.close();
   window.mainframe.tvframe.document.write(tvframe_doc);
   window.mainframe.tvframe.document.close();
}


function count_triggers() {
   ntrigs = ntrigs + 1;
   if (window.mainframe.length > 1) {
     window.mainframe.frame2.document.write(frame2_doc);
     window.mainframe.frame2.document.close();
   };
}

function get_var_in_cookie(var_name)
{
  var v, pos, start, end;
  v = var_name + "=";
  pos = document.cookie.indexOf(v);
  if (pos != -1) {
          start = pos + v.length;  //start of save_state value
          end = document.cookie.indexOf(";", start);  // end of save_state value
          if (end == -1) end = document.cookie.length;
          v = document.cookie.substring(start, end);
          return unescape(v);
  } else return "undefined";
}

function save_cookies()
{
var ExpireDate = new Date();
var expiredays = 7;

ExpireDate.setTime(ExpireDate.getTime() + (expiredays * 24 * 3600 * 1000));

document.cookie = "ntrigs=" + ntrigs + "; expires=" + ExpireDate.toGMTString();
document.cookie = "itv=" + window.mainframe.document.title + "; expires="
```

```
                    + ExpireDate.toGMTString();

}
//        load up variables

temp = get_var_in_cookie("ntrigs");
        if(temp == "undefined") {ntrigs = 0;} else {ntrigs = new Number(temp);};
itv = get_var_in_cookie("itv"); if(itv == "undefined") {itv = "main_itvframe_doc";};

</script>
</head>

<frameset onLoad="load_mainframe(itv)" onUnload="save_cookies()">
<frame name=mainframe >
</frameset>
</html>
```

7.4 Single document frameset using javascript:

The single document frameset using javascript: example has all of the features and goals of the single document frameset using document.write( ) example. However, the single document frameset using javascript: example achieves the same effect by means of using the javascript: URI scheme instead of document.write( ).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
       "http://www.w3.org/TR/REC-html40/frameset.dtd">

<html>
<head><title>DDE-1 Single Document Frameset Example using 'javascript:'</title>

<object type="application/tve-trigger" id="triggerReceiverObj" > </object>

<script type="text/ecmascript">

doctype_hdr_f = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Frameset//EN'" +
              "'http://www.w3.org/TR/REC-html40/frameset.dtd'>";

doctype_hdr_t = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'" +
              "'http://www.w3.org/TR/REC-html40/transitional.dtd'>";

main_itvframe_doc = doctype_hdr_f;
main_itvframe_doc = main_itvframe_doc +
                "<html><head><title>main_itvframe_doc<\/title><\/head>";
main_itvframe_doc = main_itvframe_doc +
              "<frameset rows='10%,10%,75%' >";
main_itvframe_doc = main_itvframe_doc +
  "<frame name=frame1 src='javascript:top.frame1_doc'>";
main_itvframe_doc = main_itvframe_doc +
"<frame name=frame2 src='javascript:top.frame2_beg_doc+top.ntrigs+top.frame2_end_doc'>";
main_itvframe_doc = main_itvframe_doc +
 "<frame name=tvframe src='javascript:top.tvframe_doc'>";
main_itvframe_doc = main_itvframe_doc + "<\/frameset> <\/html>";

main_tvframe_doc = doctype_hdr_t;
main_tvframe_doc = main_tvframe_doc +
```

```
            "<html><head><title>main_tvframe_doc<\/title><\/head>";
main_tvframe_doc = main_tvframe_doc +
                "<BODY ";
main_tvframe_doc = main_tvframe_doc +
 "style='background:url(tv:);font-size:150%;color:blue;vertical-align:top;text-align:right'>";
main_tvframe_doc = main_tvframe_doc +
       "<a href='javascript:top.main_itvframe_doc' target='mainframe'>";
main_tvframe_doc = main_tvframe_doc + "=>I<\/a>";
main_tvframe_doc = main_tvframe_doc + "<\/BODY><\/HTML>";

frame1_doc = doctype_hdr_t;
frame1_doc = frame1_doc +
          "<html><head><title>frame1_doc<\/title><\/head><body bgcolor='black' >";
frame1_doc = frame1_doc +
       "<center><a href='javascript:top.count_triggers()'>";
frame1_doc = frame1_doc + "click here to show trigger arrival effect<\/a>";
frame1_doc = frame1_doc + "<\/center><\/body><\/html>";

frame2_beg_doc = doctype_hdr_t;
frame2_beg_doc = frame2_beg_doc +
      "<html><head><title>frame2_doc<\/title><\/head><body><center><h2>";

frame2_end_doc = " triggers received<\/h2><\/center><\/body><\/html>";

tvframe_doc = doctype_hdr_t;
tvframe_doc = tvframe_doc +
    "<html><head><title>tvframe_doc<\/title><\/head><body bgcolor='black'><center>";
tvframe_doc = tvframe_doc +
 "<table><tr><td><img src='tv:' alt='TV Picture' width=320 height=240><br><\/td><\/tr>";
tvframe_doc = tvframe_doc + "<tr><td>";
tvframe_doc = tvframe_doc +
 "<a href='javascript:top.main_tvframe_doc' target='mainframe'>Full Screen<\/a>";
tvframe_doc = tvframe_doc + "-----------";
tvframe_doc = tvframe_doc +
       "<a href='tv:' target='_top'>Exit Enhancement<\/a><\/td><\/tr>";
tvframe_doc = tvframe_doc + "<\/table><\/center><\/body><\/html>";

function count_triggers() {
   ntrigs = ntrigs + 1;
   if (window.mainframe.length > 1) {
      window.mainframe.frame2.location.href =
            "javascript:top.frame2_beg_doc + top.ntrigs + top.frame2_end_doc";
   };
}

function get_var_in_cookie(var_name)
{
  var v, pos, start, end;
  v = var_name + "=";
  pos = document.cookie.indexOf(v);
  if (pos != -1) {
          start = pos + v.length;  //start of save_state value
          end = document.cookie.indexOf(";", start);  // end of save_state value
          if (end == -1) end = document.cookie.length;
          v = document.cookie.substring(start, end);
          return unescape(v);
```

```
  } else return "undefined";
}

function save_cookies()
{
var ExpireDate = new Date();
var expiredays = 7;
ExpireDate.setTime(ExpireDate.getTime() + (expiredays * 24 * 3600 * 1000));
document.cookie = "ntrigs=" + ntrigs + "; expires=" + ExpireDate.toGMTString();
document.cookie = "itv=" + window.mainframe.document.title + "; expires="
                                    + ExpireDate.toGMTString();
}
//        load up variables

temp = get_var_in_cookie("ntrigs");
        if(temp == "undefined") {ntrigs = 0;} else {ntrigs = new Number(temp);};
itv = get_var_in_cookie("itv"); if(itv == "undefined") {itv = "main_itvframe_doc";};
</script>
</head>

<frameset onUnload="save_cookies()">
<frame name=mainframe src="javascript: (top.itv == 'main_itvframe_doc') ?
                    top.main_itvframe_doc : top.main_tvframe_doc;" >
</frameset>
</html>
```

8 Acronyms

ATVEF: Advanced Television Enhancement Forum

CSS: Cascading Style Sheets

DDE: Declarative Data Essence

DOM: Document Object Model

ECMA: European Computer Manufacturers Association

GUID: Global Unique IDentifier

HTML: HyperText Markup Language

HTTP: HyperText Transport Protocol

IETF: Internet Engineering Task Force

IP: Internet Protocol

IPM: IP Multicast

MAC: Media Access Control

PCM: Pulse Code Modulation

UHTTP: Unidirectional HyperText Transport Protocol

URL: Universal Resource Locator

UUID: Universal Unique IDentifier

W3C: World Wide Web Consortium

Annex A (informative)
The javascript: URI scheme

The special “javascript:” URI scheme may be used in some contexts where a URI is permitted and where the result of accessing the resource associated with the URI may produce content of the media type text/html. Specifically, it may be used in the following contexts:

- A element, HREF attribute
- AREA element, HREF attribute
- FRAME element, SRC attribute
- IFRAME element, SRC attribute

The behavior of "javascript:" in any other contexts is unspecified.

The “javascript:” URI scheme has the following syntax:

javascriptUri : "javascript:" statementList

where the construct *statementList* shall be non-empty and shall adhere to the syntax prescribed by ISO 16262 (ECMAScript 2nd Ed.), section 12.1. Furthermore, the scheme-specific-part of a *javascriptUri* (i.e., the *statementList*) shall satisfy the syntax of the opaque_part non-terminal of the generic URI syntax prescribed by RFC 2396.

The resolution of the "javascript:" URI shall adhere to the following procedure or a logical equivalent thereof:

1. Evaluate *statementList* in an execution context identical to that used to evaluate *Global Code*, as further described by ISO 16262, section 10.2.1;

2. If the resulting value produced by (1) is not the *undefined* value, then convert the resulting value to the *String Type*; if the resulting string value is non-empty and complete, valid, DDE-1 document of media type text/html, use the resulting string value as the content of the resource accessed by this URI;

3. If the resulting value produced by (1) is the *undefined* value take no further action.

NOTES

1 If the resulting string is non-empty and invalid DDE-1 content of type text/html, the result of accessing the resource associated with the URI is unspecified.

2 The form of the "javascript:" URI which does not specify a statement list after the scheme component is not supported by DDE-1; in certain legacy browsers, such a URI, when accessed through an A element, would cause a Javascript interpreter user interface to be presented to the end-user to permit the direct entry of Javascript statements. This feature is not supported by DDE-1.

3 If any invocation of document.write()within *statementList* would modify the document which evaluates the *javascriptUri*, i.e., the document in which "javascript:" statementList is located, then the result of accessing the resource associated with the URI is unspecified.

Annex B (informative)
Bibliography

SMPTE 343M-2002, Television — Declarative Data Essence — Local Identifier (lid:) URI Scheme

SMPTE 357M-2002, Television — Declarative Data Essence — IP Multicast Encapsulation

SMPTE 361M-2002, Television — NTSC IP and Trigger Binding to VBI

SMPTE 363M-2002, Television — Declarative Data Essence — Content Level 1

SMPTE 364M, Television — Declarative Data Essence — Unidirectional Hypertext Transport Protocol

SMPTE 366M-2002, Television — Document Object Model Level 0 (DOM-0) and Related Object Environment

SMPTE RP 27.3-1989, Specifications for Safe Action and Safe Title Areas Test Pattern for 4:3 Aspect Ratio Television Systems

SMPTE-EBU Task Force for Harmonized Standards for the Exchange of Program Material as Bitstreams, Final Report: Analyses and Results, July 1998

Advanced Television Enhancement Forum (ATVEF) Specification, Draft, Version 1.1r26, updated 02/02/99, http://www.atvef.com.

IETF RFC 2519, HTTP Extensions for Distributed Authoring — WEBDAV

IETF RFC 2616, Hypertext Transfer Protocol — HTTP/1.1

IETF RFC 2838, Uniform Resource Identifiers for Television Broadcasts

IETF RFC 2965, HTTP State Management Mechanism

W3C Recommendation, HTML 4.0 Specification