100/398/PAS

DRAFT PUBLICLY AVAILABLE SPECIFICATION

Project number **62292**	
IEC/TC or SC **100**	Secretariat **Netherlands**
Distributed on **2001-07-20**	Voting terminates on **2001-09-21**
Also of interest to the following committees	Supersedes document

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

RECIPIENTS OF THIS DOCUMENT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

Title

Draft SMPTE Engineering Guideline SMPTE XXXX - Declarative Data Essence

Note:

The IEC has received from SMPTE a copyright release notice allowing the IEC to process this document as a PAS.

Draft SMPTE Engineering Guideline	SMPTE XXXX	First Draft
Declarative Data Essence		26-April- 2001

1 Scope

This document provides an overview of the Declarative Data Essence Standards, and describes how the various documents and technical components are related.

2 References and Organization

2.1 Normative References

The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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.

[DDE-1] SMPTE Proposed Standard 363M, "Declarative Data Essence, Content Level 1".
[DOM-0] SMPTE Draft Standard xxxx, "Document Object Model Level 0 (DOM-0) and Related Object Environment".
[LID] SMPTE Draft Standard xxxx, "The Local Identifier (lid:) URI Scheme".
[IPM] SMPTE Proposed Standard 357M, "Declarative Data Essence, IP Multicast Encapsulation".
[UHTTP] SMPTE Proposed Standard 364M, "Declarative Data Essence - Unidirectional Hypertext Transport Protocol".
[NTSC] SMPTE Proposed Standard 361M, "NTSC IP and Trigger Binding to VBI".
[PAL] SMPTE Draft Standard xxxx, "PAL/SECAM IP and Trigger Binding to VBI (625 Line Television Systems)".

2.2 Table of contents

3 Introduction

The 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 specifcation for "1.0", and in anticipation of both lower and higher content levels, hence the shorthand, "DDE-1".

The ATVEF specification was broken into 6 separate SMPTE documents that cover the original ATVEF specification. These specifications are:

- [DDE-1] SMPTE Proposed Standard 363M, "Declarative Data Essence, Content Level 1".
- [DOM-0] SMPTE Draft Standard xxxx, "Document Object Model Level 0 (DOM-0) and Related Object Environment".
- [LID] SMPTE Draft Standard xxxx, "The Local Identifier (lid:) URI Scheme".
- [IPM] SMPTE Proposed Standard 357M, "Declarative Data Essence, IP Multicast Encapsulation".
- [UHTTP] SMPTE Proposed Standard 364M, "Declarative Data Essence - Unidirectional Hypertext Transport Protocol".
- [NTSC] SMPTE Proposed Standard 361M, "NTSC IP and Trigger Binding to VBI".

In addition, there is a new PAL/SECAM binding:

- [PAL] SMPTE Draft Standard xxxx, "PAL/SECAM IP and Trigger Binding to VBI (625 Line Television Systems)".

And, finally there is future work to be done on wrappers for this content for carriage in SMPTE KLV. The relationship of all 8 of these documents can be found in Figure 1.

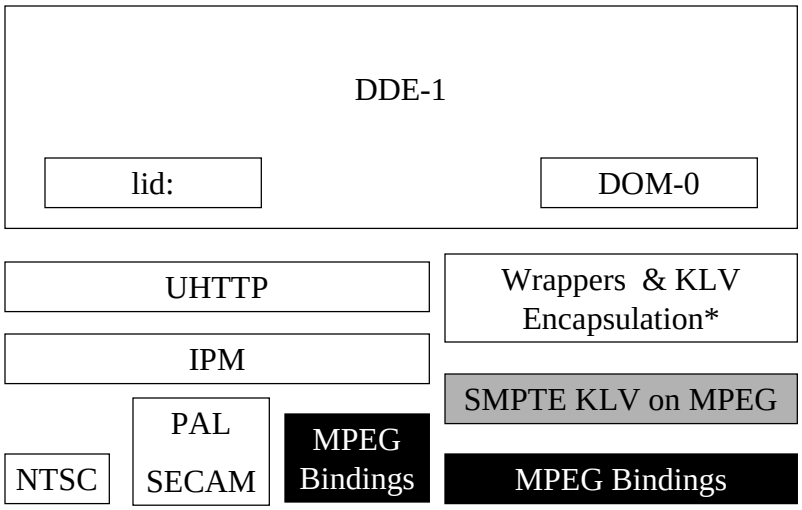


Figure 1. Relationship of the DDE-1 Standards.

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 if not overtly stated. Some more information on the expected behavior is covered in this document.

4 Receiver Behavior

[TBD]

5 Interoperability and Testing

[TBD]

6 Bibliography

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

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

Proposed SMPTE Standard SMPTE 363M	
Declarative Data Essence, Content Level 1	

1 Scope

This document defines a standard for the authoring of declarative data content intended to primarily be combined with video and/or audio services, and distributed to data-capable television signal receivers. Declarative content is generally non-procedural, and most commonly in the form of HTML. However, procedural "scripting" is also defined.

2 References and Organization

2.1 Normative References

The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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.

W3C Recommendation, "HyperText Markup Language, version 4.0"

W3C Recommendation, "Cascading Style Sheets, level 1 (CSS1)"

ISO/IEC 16262:1998, "ECMAScript language specification" [v2]

SMPTE Draft Standard xxx, "Document Object Model, Level 0 (DOM-0)"

IETF RFC 2838, "Uniform Resource Locators for Television Broadcast" [tv:]

SMPTE Draft Standard xxx, "The Local Identifier (lid:) URI Scheme" [LID]

ISO/IEC 11578:2000, Information technology, "Open Systems Interconnection – Remote Procedure Call", Annex A, "Universal Unique Identifier" [UUID]

IETF RFC 2110, "MIME E-mail Encapsulation of Aggregate Documents, such as HTML (MHTML) [text/html]

IETF RFC 2046, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" [text/plain and audio/basic]

IETF RFC 2318, "The text/css Media Type"

W3C Recommendation, "PNG (Portable Network Graphics) Specification"

ISO/IEC 10918-1, "Digital compression and coding of continuous-tone still images: Requirements and guidelines"

IETF RFC 2068, "Hypertext Transfer Protocol – HTTP/1.1"

EIA-746A, "Transport Of Internet Uniform Resource Locator (URL) Information Using TEXT-2 (T-2) Service"

ANSI/ISO 8859-1, "8-BIT SINGLE-BYTE CODED GRAPHIC CHARACTER SETS - PART 1: LATIN ALPHABET NO. 1"

Object Management Group (OMG) CORBA/IIOP 2.3.1, "The Common Object Request Broker: Architecture and Specification," Section 3, "OMG IDL Syntax and Semantics"

IANA Media Types, "application/tve-trigger", (http://www.isi.edu/in-notes/iana/assignments/media-types/media-types), "Registration of MIME media type application/tve-trigger" (http://www.isi.edu/in-notes/iana/assignments/media-types/application/tve-trigger).

IETF RFC 1952, "GZIP file format specification version 4.3"

2.2 Organization of this Document

3 Introduction

This document defines a standard for the authoring of declarative data content intended to be combined with video and audio services and distributed eventually to data-capable television receivers.

It is assumed that the reader is familiar with all the normative references and basic concepts are not discussed in this document.

This specification for enhanced television programming uses existing Internet technologies. It delivers enhanced TV programming over both analog and digital video systems using terrestrial, cable, satellite and Internet networks. The specification can be used in both one-way broadcast and two way video systems, and is designed to be compatible with all international standards for both analog and digital video systems.

A central design point was to use existing standards wherever possible and to minimize the creation of new specifications. Existing web standards, with only minimal extensions for television integration, provide a rich set of capabilities for building enhanced TV content in today's marketplace. This specification references full existing specifications for HTML, ECMAScript, DOM, CSS and media types as the basis of the content specification. The specification is not a limit on what content can be sent, but rather provides a common set of capabilities so that content developers can author content once and play on the maximum number of receivers.

Another key design goal was to provide a single solution that would work on a wide variety of networks. This design is capable of running on both analog and digital video systems as well as networks with no video at all. The specification also supports transmission across terrestrial (over the air), cable, and satellite systems as well as over the Internet. In addition, it will also bridge between networks - for example data on an analog terrestrial broadcast must easily bridge

to a digital cable system. This design goal was achieved through the definition of a transport-independent content format. This document defines two transports - one for broadcast data and one for data pulled through a return path.

While this specification has the capability to run on any video network, a complete specification requires a specific binding to each video network standard in order to ensure true interoperability. There are many roles in the production and delivery of television enhancements. This document refers to three key roles: content creator, transport operator, and receiver. The content creator originates the content components of the enhancement including graphics, layout, interaction and triggers. The transport operator runs a video delivery infrastructure (terrestrial, cable, satellite or other) that includes a transport for this data. The receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and plays this content. A particular group or company may participate as one, two or all three of these roles.

4 Content Formats

The foundation for the declarative content is existing web standards. Mandatory support is required for the following standard specifications:

?? HTML 4.0 (All 3 Document Type Definitions)

?? CSS1

?? ECMAScript

?? DOM-T

4.1 Fonts

The following 2 fonts are “default” and should be expected by authors to reside in the receiver for Latin character (8859-1) markets:

?? Tieresias, or another sans-serif font with equivalent metrics

?? monospace

The different kinds of these basic font families are defined in the following table:

Family	**Size (pixels)**	**Weight**	**Style**
Tieresias	12	Normal	Normal
Tieresias	24	Normal	Normal

Tieresias	18	Normal	Normal
Tieresias	14	Normal	Normal
Tieresias	10	Normal	Normal
Tieresias	9	Normal	Normal
Tieresias	12	Bold	Normal
Tieresias	24	Bold	Normal
Tieresias	18	Bold	Normal
Tieresias	14	Bold	Normal
Tieresias	10	Bold	Normal
Tieresias	9	Bold	Normal
Tieresias	12	Normal	Italic
Monospace	12	Normal	Normal

The CSS keyword font size table is as follows:

Font Size Name	**Height (mm)**
xx-small	4
x-small	6
small	8
medium	12
large	18
x-large	25
xx-large	40

4.2 ECMAScript Support

Authors shall assume ECMAScript support by receivers compliant with SMPTE Declarative Data Essence Transitional DOM and Object Environment. ECMAScript syntax should be both parsed and executed. All native objects and methods should be fully implemented and all language syntax and semantics supported. If authors use script objects, methods, or attributes not defined in DOM-T, such as to target a superset of DOM-T supported by a specific browser implementation, it is the author's responsibility to insert conditional logic that will only present

that script for parsing to the intended browser. Authors are advised of possible implementation limitations as follows:

?? Array sizes may be limited to 32768 elements.

?? The Number data type may be implemented as a 32-bit floating point value

?? The method, Object.prototype, may not be supported.

4.2.1 Cookie Support

Authors may count on 1KB for session cookies. Cookies support is not assumed to be persistent when a receiver is turned off.

4.3 Content Type Support

Because the architecture supports one-way broadcast of data, content creators cannot customize the content for each receiver as they do today with two-way HTTP. The following base profile of supported MIME types that that may be included in DDE content:

?? text/html (HTML 4.0)

?? text/plain

?? text/css (CSS1 only)

?? image/png (no progressive encoding)

?? image/jpg (no progressive encoding)

?? audio/basic

4.3.1 Notes on image/png

All image/png content is expected to be cached.

All possible syntax shall be permitted in the content.

All features are expected to be decoded and rendered except as follows:

?? Non square pixels

?? Gamma correction

?? Chroma correction

?? Progressive display

4.3.2 Notes on image/jpeg

All encoding formats are permitted in the content.

However, it is only expected that full JFIF (Appendix D) guidelines may be decoded and rendered. The thumbnail image may be ignored.

4.4 Triggers

Triggers are real-time events delivered for the enhanced TV program.

Note that receiver implementations may set their own policy for allowing users to turn on or off enhanced TV content, and can use trigger arrival as a signal to notify users of enhanced content availability. They are also free to decide how to turn on enhancements and how to enable the user to choose among enhancements.

Triggers always include an URL, and may optionally also include a human-readable name, an expiration date, and a script. Triggers that include a "name" attribute may be used to initiate an enhancement either automatically, or with user confirmation. If an enhancement is not currently loaded, the expiration has not been reached, and the trigger contains a "script" attribute, then the script is executed through the trigger receiver object when the page is loaded, and after all OnLoad events have fired for the enhancement. If a replaceable enhancement is currently loaded and a Trigger is received whose URL does not match the currently loaded top-level page, the Trigger expiration has not been reached, the Trigger includes a "name" attribute and a "script" attribute; then the new enhancement can be offered to the user or automatically loaded. If the top-level page of the new enhancement is loaded, the script attribute will be evaluated and executed after loading is completed. If an enhancement is currently loaded and a Trigger with a script attribute is received with a URL that matches the current top-level page, and the expiration has not been reached; the script attribute will be immediately evaluated and executed. If duplicate Triggers with script attributes are received in sequence, script attributes will be evaluated and executed for each instance. The initial top-level page for that enhancement is indicated by the URL in that trigger. Triggers that do not include a "name" attribute are not intended to initiate an enhancement, but should only be processed as events which affect (through the "script" attribute) enhancements that are currently active. If the URL matches the current top-level page, and the expiration has not been reached, the script is executed through the trigger receiver object. When testing for a match, parameters and fragment identifiers (i.e. characters in the URL including and following the first "?" or "#" character) in an URL are ignored.

Triggers are text based, and their syntax follows the basic format of the EIA-746A standard (7-bit ASCII, the high-order bit of the first byte must be "0"). Note: The triggers follow the syntax of EIA-746A, but may be transported in ways appropriate for the transport, such as multicast IP packets for example, rather than using the EIA-608 system.

All triggers are text-based and must begin with ASCII '<'. All other values for the first byte are reserved. These reserved values may be used in the future to signal additional non-text based messages. Receivers may ignore any trigger that does not begin with the '<' in the first byte.

The general format for triggers (consistent with EIA-746A) is a required URL followed by zero or more attribute/value pairs and an optional checksum:

<url> **[***attr*$_1$**:***val*$_1$**][***attr*$_2$**:***val*$_2$**]**...**[***attr*$_n$**:***val*$_n$**][***checksum***]**

The requirement to provide a checksum is transport binding specific. If required by the transport binding, the checksum must come at the end of the trigger. It is required for transport A and not required for transport B.

?? Character set: All characters are based on ISO-8859-1 character set (also known as Latin-1 and compatible with US-ASCII) in the range 0x20 and 0x7e. Any need for characters outside of this range (or excluded by attribute limits below) must be encoded using the standard Internet URL mechanism of the percent character ("%") followed by the two-digit hexadecimal value of the character in ISO-8859-1.

?? The trigger begins with a required URL:

<url>	The URL is enclosed in angle brackets (e.g. <`http://xyz.com/fun.html`>). Although any URL can be sent in this syntax, content level 1 only requires support for http: and lid: URL schemes.

The following attribute/value pairs are defined:

[name:*string***]**	The **name** attribute provides a readable text description (e.g.`[name:Find Out More]`). The *string* is any string of characters between 0x20 and 0x7e except square brackets (0x5b and 0x5d) and angle brackets (0x3c and 0x3e). The name attribute can be abbreviated as the single letter "n" (e.g.`[n:Find Out More]`).
[expires:*time***]**	The **expires** attribute provides an expiration date, after which the link is no longer valid (e.g.`[expires:19971223]`). The *time* conforms to the ISO-8601 standard, except that it is assumed to be UTC unless the time zone is specified. A recommended usage is the form *yyyymmdd*T*hhmmss*, where the capital letter "T" separates the date from the time. It is possible to shorten the *time* string by reducing the resolution. For example *yyyymmdd*T*hhmm* (no seconds specified) is valid, as is simply *yyyymmdd* (no time specified at all). When no time is specified, expiration is at the beginning of the specified day. The

	expires attribute can be abbreviated as the single letter "e" (e.g.`[e:19971223]`).
[script:*string***]**	The **script** attribute provides a script to execute within the context of the page containing the trigger receiver object (e.g.`[script:shownews()]`). The *string* is an ECMAScript statement. The form of *statement* shall be in accordance with the *StatementList* non-terminal of ECMAScript. The script attribute can be abbreviated as the single letter "s" (e.g.`[s:shownews()]`).An example of a script attribute used to navigate a frame within a page to a new URL: `[script:frame1.location.href="http://atv.com/f1"]`
[tve:*version***]**	The **`tve`** attribute indicates to the receiver that the content described in the trigger is conformant to the content specification level, *version*. For example, `[tve:1.0]`. The `"tve:"` attribute can be abbreviated as the single letter "v". The *version* can be abbreviated to a single digit when the *version* ends in ".0" (e.g.`[v:1]` is the same as `[tve:1.0]`).

The optional checksum must come at the end of the trigger. (Note: EIA-746A requires the inclusion of a checksum to ensure data integrity over line 21 bindings. In other bindings, such as IP, this may not be necessary, and is not required.)

[*checksum***]**	The checksum is provided to detect data corruption. To compute the checksum, adjacent characters in the string (starting with the left angle bracket) are paired to form 16-bit integers; if there are an odd number of characters, the final character is paired with a byte of zeros. The checksum is computed so that the one's complement of all of these 16-bit integers plus the checksum equals the 16-bit integer with all 1 bits (0 in one's complement arithmetic). This checksum is identical to that used in the Internet Protocol (described in RFC 791); further details on the computation of this checksum are given in IETF RFC 1071. This 16-bit checksum is transmitted as four hexadecimal digits in square brackets following the right square bracket of the final attribute/value pair (or following the right angle bracket if there are no attribute/value pairs). The checksum is sent in network byte order, with the most significant byte sent first. Because the checksum characters themselves (including the surrounding square brackets) are not included in the calculation of the checksum, they must be stripped from the string before the checksum is recalculated there. Characters outside the range 0x20 to 0x7e (including the second byte of two-byte control codes) shall not be included in the checksum calculation.

Other attributes could be defined at a later date. However, all other single character attribute names are set aside for future definition. Receivers may ignore attributes they do not understand.

Using the description above, all the following are valid trigger strings:

```
<http://xyz.com/fun.html>
<http://xyz.com/fun.html>[name:Find out More!]
<lid://xyz.com/fun.html>[n:Find out More!]
<lid://xyz.com/fun.html>[n:Fun!][e:19991231T115959][s:frame1.location="http://atv.com/frame1.htm"]
<http://www.newmfr.com>[name:New][tve:1][C015]
```

Note: If a trigger does not contain a `[name:]` attribute, the enhancement referenced by the trigger should not be presented to the user. Only the last example is a Trigger valid for Transport A, because it includes the [tve:] attribute and a checksum, both of which are required for Transport A.

4.4.1 More on Trigger Behavior

This section provides more description of expected trigger behavior.

Triggers may be repeated. This is commonly done to increase the chance that a Trigger will be correctly received, and to provide the opportunity to initiate an enhancement throughout a program by a viewer that joins the program in progress. However, content authors must take care that re-execution of a repeated Trigger by receivers won't break the presentation. For instance, Trigger scripts can explicitly check for whatever context or state is required before performing their function

When the tve-trigger.releasable is set to false, and a document is loaded, the following trigger shall be ignored:

`<new-URL>`*`attributes`*

However, the following trigger shall always work, independent of the state of tve-trigger.releasable:

`<current-URL>[s:window.top.location.href=<new-URL>]`*`attributes`*

More information on the expected behavior of triggers in a receiver can be found in Appendix 10.

4.5 The Trigger Receiver Object

TV enhancement HTML pages that expect to have triggers sent to them via a trigger stream shall use the HTML `object` tag to include one and only one trigger receiver object on a page. The trigger receiver object, implemented by the receiver, processes triggers for the associated enhancement in the context of the page containing the object. The content type for this object is `"application/tve-trigger"`. If a page consists of multiple frames, only one may contain a receiver object.

Sample instantiation:

```
<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">

</OBJECT>
```

Trigger Properties	Description
`enabled`	A boolean, indicating if the triggers are enabled. The default value is true (read/write)
`sourceId`	A string containing the ASCII-hex encoded UUID attribute for the announcement for this stream. `sourceId` is `null` if the `UUID` was not set for the enhancement. The UUID should uniquely identify the enhancement (for example, a different UUID for each program), and can be accessed using the trigger receiver object. In analog TV and many types of digital TV broadcast data is tied tightly to A/V. Each virtual channel has its own private network associated with it. In other systems, enhancements for many virtual channels can be carried on the same network. These systems can use the UUID to link a TV broadcast with a particular enhancement. (read only)
`releasable`	A boolean indicating that the currently displayed top-level page associated with the active enhancement can be released and may be automatically replaced with a new resource when a valid trigger containing a new URL is received. Such a trigger must contain a `[name:]` attribute. The default value is `false`. The currently displayed top-level page shall always be capable of replacing itself with a new resource.
`backChannel`	A string indicating the availability and state of a backchannel to the Internet on the current receiver. When backChannel returns "permanent" or "connected," receivers can generally perform HTTP get or post methods and expect real-time responses. When backChannel returns "disconnected," receivers can also expect to perform HTTP get or post methods but there will be an indeterminate delay while a connection is established. When backChannel returns "unavailable," no HTTP get or post methods can be performed. No standard behavior can be assumed when any other value is returned. Value is one of:

	?? `permanent`--Always connected ?? `connected`--Currently connected, but not always ?? `disconnected`--Not currently connected, but can connect ?? `unavailable`--Never connected
`contentLevel`	A number that corresponds to the content level of the receiver. For this specification, it is 1.0.

4.5.1 IDL for the application/tve-trigger Object

```
Interface Trigger {
  attribute boolean enabled;
  readonly attribute string sourceId;
  attribute boolean releasable;
  readonly attribute string backChannel;
  readonly attribute double contentLevel;
};
```


4.5.2 ECMA Script Binding

```
Object Trigger
  Properties:
    Boolean enabled
    String sourceId (readonly)
    Boolean releasable
    String backChannel (readonly)
    Number contentLevel (readonly)
};
```

4.6 URI Schemes

4.6.1 Integrating TV with Web Pages

Use the `"tv:"` URL to reference a broadcast television channel. The `"tv:"` URL may be used anywhere that a URL may reference an image.

Examples of `"tv:"` URL usage include the `object`, `img`, `body`, `frameset`, `a`, and `table` tags. For examples with specific HTML syntax, see Annex A.

Use of tv: here does not require support for the full syntax of RFC 2838, but only, literally, “tv:”.

The tv: URL can also be used for navigation, in order to make window.location.href=”tv:” and display a full screen TV image. The following example HTML code located in the top page will navigate to full screen:

Click here for full screen TV

When window.location.href==”tv:”, content authors should assume that when sending a new enhancement (i.e., an enhancement with a different URL from the one just before navigating to full screen), the new enhancement is either offered to the user, or automatically activated and displayed. In addition, the triggerReceiverObj of the immediately preceding enhancement is destroyed, just as though a navigation had occurred to a page which contained no triggerReceiverObj. This means that trigger scripts for the enhancement loaded prior to navigating to full screen are not processed.

4.6.2 The Local Identifier URL Scheme ("lid:")

Content delivered by a one-way broadcast is not necessarily available on-demand, as it is when delivered by HTTP or FTP. For such content, it is necessary to have a local name for each resource. To support cross-references within the content (for use in hyperlinks or to embed one piece of content in another), these local names must be location-independent.

The "lid:" URL scheme, as defined by [LID], enables content creators to assign unique identifiers to each resource relative to a given namespace. Thus the author can establish a new namespace for a set of content and then use simple, human-readable names for all resources within that space. The "lid:” scheme is used to identify resources that should be stored locally by a broadcast capable receiver platform and are not accessible via the Internet.

4.7 Content Caching

Content authors should expect support for one megabyte (1 MB) of cached simultaneous content. Content creators who want to reach the maximum number of receivers should manage their content to require a high-water mark of simultaneous cached content of 1 MB or less. The specific cache size required for each enhancement must be specified in the announcement.

`tve-size` represents the maximum size cache needed to hold content for the current page at any time during the program and also all pages reachable by local links. It is the high water mark during the program, not the total content delivered during the program. Size is measured as the

size when the content is delivered (after decompression for content sent using gzip or other compression techniques).

4.7.1 Cache Behavior

The content cache shall generally be expected to operate as a FIFO.

Content authors may optionally specify an HTTP Expires entity-header field for content, that gives the date/time after which the content should be considered stale. Before retrieving content out of the cache, the receiver checks the HTTP expiration time before using the content, and discards it without further use if it has expired.

All broadcast content, including CSS, shall be stored in this cache.

The cache shall be flushed on startup, but not on other conditions, including channel change.

Content that is delivered as a single entity shall not be entered into the cache until all sub-components are received.

Cache contents may be stored compressed and/or along with their transport information. But the cache size upper bound shall be measured relative to the aggregate uncompressed item sizes stripped of all transport headers.

5 Transport Types

The display of enhanced TV content consists of two steps: delivery of data resources (e.g. HTML pages) and display of named resources synchronized by triggers. All forms of transport involve data delivery and triggers. The capability of networks for one-way and/or two-way communication drives the definition of two models of transport.

This defines two kinds of transport. Transport A is for delivery of triggers by the forward path and the pulling of data by a (required) return path. Transport B is for delivery of triggers and data by the forward path where the return path is optional.

5.1 Transport Type A: Return-path Data

Most broadcast media define a way for data service text to be delivered with the video signal. In some systems, this is called closed captioning or text mode service; in other systems, this is called teletext or subtitling. For the sake of this discussion, triggers delivered over such mechanisms will be generically referred to as **broadcast data triggers**.

Some existing broadcast data services provide a mechanism for trigger delivery, but not resource delivery, due to limited bandwidth. Content creators may encode broadcast data triggers using these mechanisms. Broadcast data streams only contain broadcast data triggers so there is no announcement or broadcast content delivery mechanism. Because there are no announcements, the broadcast data service stream is considered to be implicitly announced as a permanent session.

In addition to the other attributes used in triggers, transport type A triggers must contain an additional attribute, `"tve:"`. This attribute is ignored if present in a trigger in transport B since these values are set in transport type B in the announcement. If the `"tve:"` attribute is not present in a transport type A trigger, the content described in the trigger is not considered to be DDE content.

Television transport operators should use the standard mechanisms for broadcast data trigger transmission for the appropriate medium (EIA, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver locates and delivers broadcast data triggers associated with the TV broadcast. Tuning and decoding broadcast data triggers is implementation and delivery standard specific and is specified in the appropriate binding. A mechanism must be defined for encoding broadcast data triggers for each delivery standard. Because there is no content delivery system, broadcast data triggers usually require two-way Internet connections to fetch content over HTTP, version 1.1.

Note: Television transport operators and content creators need to plan to handle the scalability issues associated with large numbers of HTTP requests responding at roughly the same time to broadcast triggers.

5.2 Transport Type B: Broadcast Data

Transport type B is for true broadcast of both the resource data and triggers. As such, transport type B can run on TV broadcast networks without Internet connections, unlike transport type A. An additional Internet connection allowing a return path can be added to provide two-way capabilities like ecommerce or general Web browsing.

Transport type B uses announcements to offer one or more enhancements of a TV channel. An announcement specifies the location of both the resource stream (the files that provide content) and the trigger stream for an enhancement. Multiple enhancements can be offered as choices that differ on characteristics like language,required cache size, or bandwidth. In addition to locating the files and trigger streams, announcements must be able to provide the following information: language, start and stop times, bandwidth, peak storage size needed for incoming resources, the content level the resources represent, an optional UUID that identifies the content, an optional string that identifies the broadcast channel for systems that send ATVEF content separately from the audio/video TV broadcast. The receiver must be able to start receiving data from only the description broadcast in the announcement.

Transport type B also requires a protocol that provides for delivery of resources. In one-way broadcast systems, this is a one-way resource transfer protocol that allows for broadcast delivery of resources. The resource delivered, no matter what the resource transfer method, must include HTTP headers to package the file on the resource transfer protocol. All resources delivered using resource transfer are named using URLs. These resources are then stored locally, and retrieved from this local storage when referenced using this same URL. Content authors should expect

support for local storage and retrieval of content using the `"lid:"` URL scheme and the familiar `"http:"` URL scheme. When `"lid:"` is used, the resources are delivered only through broadcast and are not available on demand. When "http:" is used, the resources that are delivered through broadcast also exist on the World Wide Web and can be requested from the appropriate server using standard HTTP. Sending `"http:"` resources using resource transfer effectively pre-loads the local cache, thus avoiding large numbers of simultaneous hits on Web servers when those same resources are requested by many receivers. Furthermore, this mechanism allows receivers to view the same content that appears on the Web even when no Internet connection is available. Content creators can freely mix resources that use either the `"lid:"` or `"http:"` schemes in the same enhanced broadcast. Because the underlying resource transfer protocol is not limited to carrying resources named by any particular URL scheme, some receivers will store and retrieve content named using other URL schemes, such as `"ftp:"`, as well as the required `"lid:"` and `"http:"`.

Transport type B uses the same syntax for triggers as type A.

5.3 Simultaneous Support of Transports A and B

A single video program may contain both transport type B and transport type A (e.g. broadcast data triggers) simultaneously. This is advantageous in order to target both IP-based receivers as well as receivers that can only receive broadcast data triggers.

Receivers may choose to support only Transport B based trigger streams and ignore broadcast data triggers (Transport A), or receivers may support broadcast data triggers in the absence of Transport B based triggers, or receivers may support broadcast data triggers and Transport B based triggers simultaneously. For receivers that provide simultaneous support, this specifies the following behavior, which is identical to the treatment of Transport B based triggers on an active stream.

When a broadcast data trigger is encountered, its URL is compared to the URL of the current page. If the URLs match and the trigger contains a script, the script should be executed. If the URLs match but there is no script, the trigger is considered a retransmission of the current page and should be ignored. If the URLs do not match and the trigger contains a name, the trigger is considered a new enhancement and may be offered to the viewer if triggerReceiverObj.releasable is true. If the URLs do not match and there is no name, the trigger should be ignored.

6 Annex A: Examples of Integrating TV with Web Pages (Informative)

The following examples describe how to achieve common design goals for integrating TV and Web pages. This list is meant to be illustrative rather than exhaustive. The `"tv:"` URL may be used anywhere that an image URL is also appropriate.

Examples are presented in both HTML 3.2 and HTML 4.0 since the HTML 4.0 specification recommends that tools supporting HTML 4.0 continue to support HTML 3.2.

6.1 How to place TV in a web page (using <OBJECT> and <IMG> tags)

The OBJECT and IMG tags are used to place the TV picture in a web page, for example:

```
<object data="tv:" width="60%" height="60%">
<img src="tv:" width=320 height=240>
```

6.2 How to place TV in a web page that uses tables (using <TABLE> tags)

The TD tag can be used to place the TV picture as the background of a table cell, for example:

```
<td width=320 height=240 style="background: url(tv:)">
    Here is content that is overlaid on top of the
    TV picture inside this table cell.
</td>
```


6.3 How to overlay a web page over a TV background (using <BODY> tag)

The BODY tag is used to specify TV as a full screen background of the web page, for example:

```
HTML 3.2 syntax: <body background="tv:">
HTML 4.0 syntax: <body style="background: url(tv:)">
```

6.4 How to overlay a frame-based web page over a TV background (using <FRAMESET> tag)

Many web pages will be frame-based rather than body tag based. This will allow the program to change the displayed web page while maintaining the same URL for a series of triggers. Since an HTML document that contains a FRAMESET tag cannot contain a BODY tag, it is necessary to specify `"tv:"` on a FRAMESET when full screen TV is desired beneath the frames, for example:

```
<frameset style="background: url(tv:)" cols="200,*">
```

Each frame in the frameset that wants the full screen TV to show through must specify a transparent background color in the BODY tag of the frame's HTML document, for example:

```
HTML 3.2 syntax: <body bgcolor="transparent">
HTML 4.0 syntax: <body style="background: transparent">
```

6.5 How to transition from a web page back to full-screen TV (using <A> tag)

Finally, the use of `"tv:"` as the `href` of an anchor tag allows for hyperlinking to full screen TV, for example:

```
   <a href="tv:">Click here to go to full-screen TV</a>
```

DRAFT

7 Appendix B: Using Enhanced TV (Informative)

Television enhancements delivered using Transport A only contain broadcast data triggers so there is no announcement or broadcast content delivery mechanism. Because there are no announcements, the broadcast data service stream is considered to be implicitly announced as a permanent session.

Television enhancements delivered using Transport B are comprised of three related data sources: announcements, content, and triggers.

Announcements have a time period for which they are valid, and also contain information that the client can optionally use to help decide whether to automatically start receiving trigger and content information. This may include type, language and keyword attributes that provide additional information to the client about the announced enhancements.

When the client sees a new enhancement, it knows that there will be data available on the given content and trigger addresses. The client may present the user with a choice to start receiving trigger and content information, or may do so automatically. The client implementation specifies what kind of user interface, if any, to present. After this confirmation (or automatic behavior) the client receives content and triggers, caching the content and parsing the triggers.

When the client first receives a trigger (containing a URL pointing to some enhancement content) the client may notify the user that the content is available or, alternatively, navigate to that content automatically. Clients may choose not to notify the user if they believe that they cannot display the enhancement, generally because the content referred to by the specified URL is not available.

When an enhancement has either been confirmed by the user, or has been started automatically, the enhancement is displayed. Only one enhancement may be displayed at a time. When new triggers associated with the current enhancement arrive, they are executed or ignored depending on several conditions. If the URL of the trigger matches the URL of the current page and the trigger has a script attribute, the script is played; if there is no script, the trigger is ignored. If the URLs do not match and the trigger has a name attribute, the trigger is considered a new enhancement and is played, offered to the viewer, or ignored depending on other factors described below; if no name attribute, the trigger is ignored.

If a new enhancement is announced while an existing enhancement is being displayed, the client may present the user with the option to begin receiving that announcement data (content and triggers) or do so automatically. Multiple enhancements may be received simultaneously, although only one may be displayed at a time.

When the new enhancement is being received at the same time as an existing enhancement is being displayed, and the new enhancement delivers its first trigger, the client may have one of three behaviors:

?? The client ignores the new enhancement trigger until the existing enhancement has been completed.

?? It presents the user with the opportunity to navigate to the new enhancement.

?? The client automatically navigates to the new enhancement.

It may be important for some triggers to be able to send scripts to the current enhancement without presenting the user with the opportunity to navigate to that enhancement. In this case, no `[name:]` attribute should be included. This allows enhancements to enforce that the user views them from the beginning and does not join in later when a subsequent trigger containing a script is received. If no `[name:]` attribute is found in the trigger, the user should not be presented with the opportunity to view the enhancement or automatically navigate there. The enhancement's data stream can be used to pre-load data by sending data before the first trigger that is sent with a `[name:]` attribute.

Content creators are encouraged to "shut down" their enhancements at the end of the related video content. This means that enhancements should navigate themselves (via trigger scripts or some other scripting mechanism) to full screen television (`"tv:"`) when the program or commercial ends. This will prevent content creators from displaying their enhancement over some unrelated broadcasts and reduce the likelihood of conflicts between producers. Content creators may wish to collaborate with the producers of subsequent programs or commercials to build a single enhancement that spans multiple video segments and may provide some enhanced user experience. For example, a broadcaster may provide a standard top level page that loads and displays all channel content, program content, and advertisement content in separate frames, which can be loaded and displayed at any time in response to script triggers or user initiated scripts executed in the top level page. This permits ad content to be preloaded in hidden frames while other enhancements are displayed, and the quick transition from program enhancements to ad enhancements and back again.

When the time period specified by the announcement is over, clients may automatically end the enhancement to prevent the user from viewing the enhancement over potentially unrelated video.

A property, named triggerReceiverObj.releasable may be set on the trigger receiver object associated with the current enhancement. When set to `true`, the current enhancement associated with this trigger stream may be automatically replaced with a new enhancement if the client user interface permits this. A subsequent enhancement can become active by sending a trigger which includes a `[name:]`attribute when the current page's trigger receiver object's triggerReceiverObj.releasable property is `true`.

An example showing a more complex use of triggers and pages follows.

This content would consist of two original source files, an HTML document and a PNG image. The experience consists of a screen with a 60% sized embedded live TV object, with some text below it. During the show, a trigger may arrive that will cause an image of the word "MURDER" to appear below the text. If the user chooses to click on the TV object, they will be returned to full screen video, and away from the enhanced experience

The first would be referred to by the URL:

`<lid://nicebroadcaster.com/show27/launch.html>`, and consists of the following text:

```
<HTML>
```

```
<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>


<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>


<SCRIPT LANGUAGE="JavaScript">
function scenechange(imagename)
{
document.sceneimage.src = imagename + ".png";
}
</SCRIPT>


<BODY bgcolor="magenta">


<A href="tv:">
<OBJECT data="tv:" width="60%" height="60%" align="center">
</OBJECT>
</A>


<BR>
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>

<IMG name="sceneimage" align="center" src="">

</BODY>
</HTML>
```


The second file consists of a PNG image, containing an image of the word "MURDER" in big red letters. It's URL will be specified as `<lid://nicebroadcaster.com/show27/murder.png>`

The following trigger would be sent after the data was first transmitted to trigger the beginning of the enhanced television experience:

```
<lid://nicebroadcaster.com/show27/launch.html>[name:Day & Night & Day Again Interactive]
```

This trigger packet would also be transmitted periodically later on, to allow viewers who tune in late to join in the fun.

Later on, during the program, the content creator might send the following trigger to make the content change to reflect the fact that a murder scene has just begun in the program:

```
<lid://nicebroadcaster.com/show27/launch.html>[script:scenechange("murder")]
```

This trigger would cause the active enhancement page (if it matched the URL in the trigger) to execute the ECMAScript function 'scenechange("murder")', which would cause the murder.png image to be displayed within the page. If the specified URL was not currently being displayed, the trigger would be ignored because this trigger does not include a [name:] attribute,

Near the end of their program, they might send the following trigger to tell their interactive application to shut down. This would allow them to more accurately synchronize with the end of the program, rather than relying on the session timing information in the announcement.

```
<lid://nicebroadcaster.com/show27/launch.html>[script:window.location="tv:"]
```

8 Appendix C: Glossary (Informative)

Announcements: Announcements are used to announce currently available programming to the receiver.

Backchannel: A connection from a receiver to the Internet or back to some server.

Client: The receiver environment that renders the content.

Content creator: In the context of this document, a content creator has the role of originating the content components of the television enhancement including graphics, layout, interaction, and triggers.

CSS1 (Cascading Style Sheets, Level 1): CSS1 is a simple style sheet mechanism that allows content creators and readers to attach style (e.g. fonts, colors and spacing) to HTML documents. The CSS1 language is human readable and writable, and expresses style in common desktop publishing terminology.

DOM (Document Object Model): the Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

ECMA Script: A general purpose, cross-platform programming language.

Enhancement: This content added to a video/audio service.

Forward Path: The television broadcast stream.

HTML (Hypertext Markup Language): a collection of tags typically used in the development of Web pages.

HTTP (Hypertext Transfer Protocol): a set of instructions for communication between a server and a World Wide Web client.

IETF (Internet Engineering Task Force): the IETF is a large, open community of network designers, operators, vendors, and researchers whose purpose is to coordinate the operation, management and evolution of the Internet, and to resolve short-range and midrange protocol and architectural issues. It is a major source of proposals for protocol standards which are submitted to the IAB for final approval. The IETF meets three times a year and extensive minutes are included in the IETF Proceedings.

ISO (International Organization for Standardization): a voluntary, non treaty organization founded in 1946 which is responsible for creating international standards in many areas, including computers and communications. Its members are the national standards organizations of the 89 member countries, including ANSI for the U.S.

MIME (multipart/signed, multipart/encrypted content-types) a protocol for allowing e-mail messages to contain various types of media (text, audio, video, images, etc.).

NABTS (North American Basic Teletext Specification).

Receiver: In the context of this document, a receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and presents declarative content.

Return Path: See backchannel.

Triggers: used to identify the URL and some human-readable string to use in the announcement to the user. In order to announce the availability of the interactive television experience to the user, (as opposed to announcing it to the client downloader mechanism).

TV Enhancement: A collection of Web content displayed in conjunction with a TV broadcast as an enhanced or interactive program.

UUID (Universally Unique Identifier) Also known as GUID (Globally Unique IDentifier) is an identifier that is unique across both space and time, with respect to the space of all UUIDs.

W3C (World Wide Web Consortium): The W3C, an an international industry consortium, was founded in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability.

9 Appendix D JFIF Guidelines for JPEG (Normative)

This Appendix further restricts the JPEG encoding and defines a new APP0 field defining a profile commonly known as JFIF. The syntax of a JFIF-profiled JPEG file conforms to the syntax for interchange format defined in Annex B of ISO DIS 10918-1.

The encoding restrictions are:

?? Baseline compression only.

?? The only color space is YCbCr as defined by CCIR 601 (normalized to 256 levels). The resulting RGB components shall not be gamma corrected. If only one component is used, that component shall be Y.

?? The image orientation is always top-down.

?? The position of the pixels in sub-sampled components are defined with respect to the highest resolution component.

The following APP0 marker is mandatory right after the SOI marker, and is defined as follows:

Field	Size (bytes)	Value
APP0	1	JPEG APP0 code
Length	2	Length of the structure including this field
Identifier	5	"JFIF" with null termination and zero parity
Version	2	0x0102
Units	1	The units for (Xdensity, Ydensity) 0x00=aspect ratio 0x01=dots per inch 0x02=dots per cm
Xdensity	2	Horizontal pixel density
Ydensity	2	Vertical pixel density
Xthumbnail	1	Horizontal thumbnail pixel density
Ythumbnail	1	Vertical thumbnail pixel density
RGB	3n	Thumbnail RGB values packed in 24 bits, where n=Xthumbnail*Ythumbnail

10 Appendix E – Notes on Receiver Trigger Behavior (Informative)

The behavior associated with the receipt of a trigger depends on a combination of factors relating to the content of the trigger and the state of the currently loaded enhancement (if any). The behavior is described in the two tables below for the following cases:

?? **Table E.1** – triggers received when no enhancement is currently loaded

?? **Table E.2** – triggers received when an enhancement is currently loaded

For each of these tables the following preconditions apply:

?? If the transport (i.e., Transport A or B) and/or binding requires the use of a checksum, then the tables assume that the checksum is valid. If a checksum is present and is not valid, then the trigger is ignored.

?? If the transport and/or binding requires that an announcement is received and processed before a trigger associated with that announcement is accepted, then the tables assume that such an announcement has been received and processed. For such transports and/or bindings, if the announcement for which the trigger is associated has not been received and processed, then the trigger is ignored.

?? If the trigger contains an expiration date, then the tables assume that the expiration date has not yet been reached. If the expiration date has already been reached, then the trigger is ignored.

10.1 Table E.1 – No enhancement currently loaded

If no enhancement is currently loaded, the following table describes receiver behavior as a function of the contents of a received trigger:

Trigger Contains name	Trigger Contains script	Receiver Action
no	no	Ignore trigger.
no	yes	Ignore trigger.
Yes	no	Process enhancement.[1]
Yes	yes	Process enhancement[1] and execute script**Error! Bookmark not defined.**.

Table E.1

[1] Receiver action with regard to the enhancement is imp lementation specific. It is expected that the receiver would have one of the following behaviors:

1. The enhancement is offered to the user, i.e., the user is presented with the opportunity to activate the enhancement. If the user activates the enhancement, the page is displayed.
2. The enhancement is automatically activated and displayed .

10.2 Table E.2 – An enhancement is currently loaded

If an enhancement is currently loaded the following conditions describe receiver behavior as a function of the contents of a received trigger:

If the triggerReceiverObj.releasable property of the current tve-trigger receiver object is set to false:

1. All triggers of the form:

   `<new-URL>`*`attributes`*

   `shall be ignored.`

2. `All triggers of the form:`

   `<current-URL>[s:window.top.location.href=`
   `<new-URL>]`*`attributes`*

   shall be processed in accordance with the Table E.2.The current enhancement shall always be capable of navigating to a new URL regardless of the state of tve-trigger.releasable.

If the tve-trigger property triggerReceiverObj.releasable of the currently loaded enhancement is set to true then all triggers are processed in accordance with Table E.2.

Trigger URL= Current Page URL	Trigger Contains name	Trigger Contains script	Receiver Action
no	no	no	Ignore trigger.
no	no	yes	Ignore trigger.
no	yes	no	Process new enhancement.[2]
no	yes	yes	Process new enhancement[2] and execute script.[3]
yes	no	no	Ignore trigger.
yes	no	yes	Execute script in the context of the current document.
yes	yes	no	Ignore (retransmission of current page).
yes	yes	yes	Execute script in the context of the current document.

[2] Receiver action with regard to the new enhancement is implementation specific. It is expected that the receiver would have one of the following behaviors:

1. The new enhancement is ignored until the existing enhancement has been completed, i.e., the trigger is queued.
2. The new enhancement is offered to the user, i.e., the user is presented with the opportunity to activate the new enhancement. If the user activates the new enhancement, the new page is displayed. Only one enhancement shall be active at a time.
3. The new enhancement is automatically activated and displayed. Only one enhancement shall be active at a time.

[3] Footnote 3 is applied and when the new enhancement is activated, the script shall be executed in the context of the new document.

11 Appendix F - Security Model (Informative)

This system assumes that **all** insertion points of the content described in this document are trusted sources (or "hosts" in Internet thinking). Therefore, the use of scripts in triggers, and ECMA Script in general is not presumed to be a security problem for malicious content or viruses any more than the notion of roguevideo and audio being inserted into a broadcaster's emission. As with video and audio, the broadcaster is responsible for the quality of the data content of its programming.

The security ramifications of an application of this specification where the insertion points are not all trusted has not been studied and is unknown.

Draft SMPTE Standard SMPTE XXXX **Document Object Model Level 0 (DOM-0) and Related Object Environment**	Draft r 20-April-2001

1 Scope

This document defines a document object model (DOM) and an object environment for use in manipulating HTML documents using the ECMAScript environment of Declarative Data Essence. This standard reflects the best current practice for continuing use in television and other applications..

2 References and Organization

2.1 Normative References

ISO/IEC 16262:1998, "ECMAScript language specification" [ECMAScript, Edition 2]

2.2 Informative References

W3C Recommendation, "Document Object Model (DOM) Level 1 Specification"

Netscape, Javascript 1.3 [Product Documentation]

Microsoft, JScript [Product Documentation]

Microsoft, Javascript Host DOM Objects [Product Documentation]

Object Management Group (OMG) CORBA/IIOP 2.3.1, "The Common Object Request Broker: Architecture and Specificatin", Section 3, "OMG IDL Syntax and Semantics"

SMPTE Proposed Standard 363, "Declarative Data Essence, Content Level 1"

2.3 Document Organization

3 Introduction

This is a definition for a Document Object Model and Object Environment (DOM-0), to allow orderly transition from existing current authoring practice to DOM-1 and DOM-2. DOM-0 is in widespread use today in the form of the objects and methods implemented by the popular browser vendors. This is an attempt to document something like DOM-0 (as was loosely defined by W3C). Its provides an authoring standard to those that wish to make content that will be compatible with the largest number of "down level" browsers, such as some used in interactive television receivers supporting the Declarative Data Essence Content Level 1 specification.

Standardized support for a DOM-0 is therefore required if one expects the existing tools, content, and author knowledge-base to be usable in the short term. And, DOM-0 support in browsers will continue to live on well beyond deployment of DOM-2, in order to not require re-authoring of everything on the web.

4 Definition (with IDL)

This document is intended to define the intersection of Microsoft Internet Explorer, version 3.0 (IE3) and Netscape Navigator, version 3.0 (NN3) relative to their DOM.
Additionally, there are some important features of both IE3 and NN3 that are not in ECMAScript, but not strictly part of the DOM either. These must also be addressed. So DOM-0 is a baseline DOM, as well as a set of miscellaneous functions including an extension to the ECMAScript String object, and the addition of some navigation objects. Collectively, this is referred to as DOM-0.
.
The definition here is broken into three main groups:

- ?? DOM
- ?? ECMA Script Extensions
- ?? Navigation

The DOM is the part that maps to a strict definition of a Document Object Model (and is in theory easily mappable onto DOM-1). The ECMA Script Extensions are objects or methods that extend the definition of standard ECMA Script objects to handle current practice. And, navigation is a set of objects that handle some of the common functions that have come to be expected in a given implementation of the client. These latter 2 categories are not DOM, but supporting objects that have become common for authors to use.

The strict DOM consists of the following objects:

- ?? Anchor
- ?? Button
- ?? Checkbox
- ?? Document
- ?? Form
- ?? Hidden
- ?? Image
- ?? Link
- ?? Location
- ?? Option
- ?? Password
- ?? Radio
- ?? Reset
- ?? Select
- ?? Submit
- ?? Text
- ?? TextArea

The ECMA Script object extension is an extension of the standard ECMA Script:

- ?? String

The new Navigation objects are:

- ?? History
- ?? Navigator
- ?? Window

These are all defined as a flat object space. There is no inheritance or hierarchy and all methods are fully defined in each object definition.

Follows is the IDL for each object.

4.1 Anchor

```
Interface Anchor {
  attribute String name;
};
```

An HTML anchor tag object.

Properties

?? name - the HTML NAME attribute

Methods

None

4.2 Button

```
Interface Button {
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void click();
};
```


An HTML form button object.

Event Handlers

?? onBlur
?? onClick
?? onFocus

Properties

?? value - the string on the face of the button

Methods

?? blur()
?? click()

4.3 Checkbox

```
Interface Checkbox {
  attribute boolean checked;
  readonly attribute boolean defaultChecked;
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void click();
  void focus();
};
```

This is the HTML form checkbox object.

Event Handlers

- ?? onBlur
- ?? onClick
- ?? onFocus

Properties

- ?? checked - boolean state of the checkbox (true if currently checked, else false)
- ?? defaultChecked - the HTML CHECKED attribute
- ?? form – the Form containing this element
- ?? name - the HTML NAME attribute
- ?? type – the type of this form element
- ?? value - the HTML TYPE attribute

Methods

- ?? blur()
- ?? click()
- ?? focus()

4.4 Document

```
Interface AnchorSequence {
  Anchor item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface FormSequence {
  Form item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface ImageSequence {
  Image item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface LinkSequence {
  Link item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface Document {
  attribute String alinkColor;
  readonly attribute AnchorSequence anchors;
  attribute String bgColor;
  attribute String cookie;
  attribute String fgColor;
  readonly attribute FormSequence forms;
  readonly attribute ImageSequence images;
  attribute String lastModified;
  attribute String linkColor;
  readonly attribute LinkSequence links;
  readonly attribute String location;
  readonly attribute String referrer;
  readonly attribute String title;
  readonly attribute String URL;
  attribute String vlinkColor;
  void clear();
  void close();
```

```
  void open(in String mimeType);
  void write(in String expr1, in String expr2, ...);
  void writeln(in String expr1, in String expr2, ...);
};
```

The document object.

Properties

- alinkColor - the HTML ALINK attribute. Note that the leading # is optional for all numeric values, and the Standard Color Names are supported.
- anchors[] - an array of anchor objects in the order the anchor tags appear in the HTML document. Use anchors.length to get the number of anchors in a document.
- bgColor - the HTML BGCOLOR attribute. Note that the leading # is optional for all numeric values, and the Standard Color Names are supported.
- cookie - all the cookie strings sent in the HTTP reply headers currently saved and concatenated. The syntax for each one is name=value;expires=expDate; The expDate format is: "Wdy, DD-Mmm-YY HH:MM:SS GMT".
- fgColor - the HTML FGCOLOR attribute. Note that the leading # is optional for all numeric values, and the Standard Color Names are supported.
- forms[] - an array of form objects in the order the forms appear in the HTML file. Use forms.length to get the number of forms in a document. Note that only the formal array form is permitted (i.e. forms[index]).
- images[] - an array of image objects in the order they appear in the HTML document. Use images.length to get the number of images in a document.
- lastModified - the value of the HTTP reply header field of the same name. Date formats are as permitted by HTTP 1.1.
- linkColor - the HTML LINK attribute. Note that the leading # is optional for all numeric values, and the Standard Color Names are supported.
- links[] – an array of link objects in the order the hypertext links appear in the HTML document. Use links.length to get the number of links in a document.
- location - a deprecated synonym for the URL property. This property exists only for compatibility with JavaScript 1.0.
- referrer – string that contains the URL of the document that this was linked from.
- title - the HTML TITLE tag
- URL – a read-only string reflecting the actual URL of this document. Note that document.URL may differ from window.location because the actual URL of the document may be different from the URL used to request the document. Document.URL is not reflected in the window's location object.
- vlinkColor - the HTML VLINK attribute. Note that the leading # is optional for all numeric values, and the Standard Color Names are supported.

Methods

- clear() – clear the contents of this document.
- close() - close a previously opened new document and display it.
- open() - open a new document for creation.
  - mimeType - the ContentType of the document.
- write() - output one or more expressions to the currently open document.
  - exprn - any ECMAScript expression that results in a string.
- writeln() - same as write() except a newline is inserted at the end of all the expressions.
  - exprn - any ECMAScript expression that results in a string.

4.5 Form

```
Interface ObjectSequence {
  Object item(in unsigned long index);
  readonly attribute unsigned long length;
```

```
};
Interface Form {
  attribute String action;
  readonly attribute ObjectSequence elements;
  attribute String encoding;
  readonly attribute long length;
  attribute String method;
  attribute String name;

  attribute String target;
  void submit();
};
```

The HTML Form object.

Event Handlers

- ?? onSubmit

Properties

- ?? action - the HTML ACTION attribute.
- ?? elements[] - an array of objects for each form element in the order in which they appear in the form.
- ?? encoding – the string value of the MIME encoding as specified in ENCTYPE attribute
- ?? length - number of elements in "elements" above.
- ?? method - the HTML METHOD attribute.
- ?? name – the name of the form
- ?? target - the HTML TARGET attribute.

Methods

- ?? submit() - cause the form to be submitted.

4.6 Hidden

```
Interface Hidden {
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
};
```

The HTML Form Hidden field object.

Properties

- ?? form – the Form containing this element
- ?? name – the name of this form element
- ?? type – the type of this form element
- ?? value - the element value.

Methods

- ?? *<none>*

4.7 History

```
Interface History {
  void back();
  void forward();
  void go(in long delta);
  void go(in String location);
};
```

This provides a global history list and management. Note that it is a property of the Window Object , and not per frame. Document URL's are added to the history whenever a link is taken by the user. Whenever one navigates "back" and then a peer branch is taken, the other peer tree (if any) of document URL's is pruned from the history. Navigating "back" causes the document that the user navigated from to reach the current document to be displayed. Navigating "forward" causes the document most recently reached from the current document to be displayed. The history list includes the navigation from triggers just like they were links that the user had taken.

When matching with go(), the **location** argument is matched against URLs in the history list and a partial match occurs when the full string content of **location** matches the initial set of characters in one of the URLs (i.e. the match is anchored to the beginning of the URL). This is started at the current position in the forward direction, wrapping around to the top.

Properties:

- ?? *<none>*

Methods:

- ?? back() - load the previous URL in the history list.
- ?? forward() - load the next URL in the history list.
- ?? go() - load a specific URL in the history list.
  - ?? delta - an integer number of an item in the history list relative to the current document. A positive number indicates the equivalent of "delta" forward() calls. A negative number indicates the equivalent of "delta" back() calls.
  - ?? location - a URL of an item in the history list. Partial matches are permitted where the first partial string match is loaded.

4.8 Image

```
Interface Image {
  attribute long border;
  readonly attribute boolean complete;
  attribute long height;
  attribute long hspace;
  attribute String lowsrc;
  attribute String name;
  attribute String src;
  attribute long vspace;
  attribute long width;
};
```

The image object reflects an image included in an HTML document.

Event Handlers

- ?? onLoad

Properties

- ?? border – an integer value reflecting the width of the image's border in pixels.
- ?? complete - a boolean value indicating whether the image has finished loading.

- ?? height – an integer value reflecting the height of an image in pixels.
- ?? hspace – an integer value reflecting the HSPACE attribute of the <IMG> tag, which controls the amount of white space, in pixels, to the left and right of the image.
- ?? lowsrc – a string value containing the URL of the low-resolution version of the image to load.
- ?? name – a string value indicating the name of the image object.
- ?? src – a string value indicating the URL of the image.
- ?? vspace – an integer value reflecting the VSPACE attribute of the <IMG> tag, which controls the amount of white space, in pixels, above and below the image.
- ?? width – an integer value indicating the width of an image in pixels.

4.9 Link

```
Interface Link {
  attribute String hash;
  attribute String host;
  attribute String hostname;
  attribute String href;
  attribute String pathname;
  attribute String port;
  attribute String protocol;
  attribute String search;
  attribute String target;
};
```


The HTML Link object of the form:

protocol://host:port/pathname?search#hash

Event Handlers

- ?? onClick
- ?? onMouseOut
- ?? onMouseOver

Properties

- ?? hash - the URL fragment component.
- ?? host - the URL domain name (or IP address) part of the authority component.
- ?? hostname - the entire authority component, including port number, if present.
- ?? href - the entire URL.
- ?? pathname - the URL path component.
- ?? port - the port number of the URL authority component. Zero if none.
- ?? protocol - the scheme name, including the ":"
- ?? search - the URL query component.
- ?? target - the HTML TARGET attribute.

Methods

- ?? *<none>*

4.10 Location

```
Interface Location {
  attribute String hash;
  attribute String host;
```

```
  attribute String hostname;
  attribute String href;
  attribute String pathname;
  attribute String port;
  attribute String protocol;
  attribute String search;
};
```

The location object of a window is a reference to Location object, which is a representation of the URL of the document currently being displayed in that window. Location represents the URL used to request the current document, but may not equal the actual URL of the document, which is stored in the string document.URL. In addition to its properties, the Location object can be used as if it were itself a primitive string value. If you read the value of a Location object, you get the same string as you would if you read the href property of the object. Assigning a URL to the location property of a window, causes the browser to load and display the contents of the URL assigned to it.

Properties

- ?? hash - the URL fragment component.
- ?? host - the URL domain name (or IP address) part of the authority component.
- ?? hostname - the entire authority component, including port number if present.
- ?? href - the entire URL.
- ?? pathname - the URL path component.
- ?? port - the port number ofhte URL authority component. Zero if none.
- ?? protocol - the scheme name, including the ":"
- ?? search - the URL query component.

Methods

- ?? *<none>*

4.11 Navigator

```
Interface Navigator {
  readonly attribute String appCodeName;
  readonly attribute String appName;
  readonly attribute String appVersion;
  readonly attribute String userAgent;
};
```

Properties:

- ?? appCodeName - the code name of the browser imp lementation.
- ?? appName - the browser vender.
- ?? appVersion- the browser version number.
- ?? userAgent - the HTTP Request Header UserAgent field value.

Methods

- ?? *<none>*

4.12 Option

```
Interface Option {
  readonly attribute boolean defaultSelected;
  readonly attribute long index;
  attribute boolean selected;
```

```
  readonly attribute String text;
  attribute String value;
};
```

The HTML Form Option object.

Properties

- ?? defaultSelected - initial state of the option selection.
- ?? index –an integer value specifying the position of the option in the select list.
- ?? selected - the current selection state.
- ?? text - the text of an option in the selection list.
- ?? value - the value returned to the server.

Methods

- ?? *<none>*

4.13 Password

```
Interface Password {
  readonly attribute String defaultValue;
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void focus();
  void select();
};
```


The HTML Form Password object.

Event Handlers

- ?? onBlur
- ?? onFocus

Properties

- ?? defaultValue - the HTML VALUE attribute.
- ?? form – the Form containing this element
- ?? name – the name of this form element
- ?? type - the HTML TYPE attribute.
- ?? value - the current value of the field.

Methods

- ?? blur()
- ?? focus()
- ?? select()

4.14 Radio

```
Interface Radio {
  attribute boolean checked;
  readonly attribute boolean defaultChecked;
```

```
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void click();
  void focus();
};
```

The HTML Form Radio object.

Event Handlers
- ?? onBlur
- ?? onClick
- ?? onFocus

Properties
- ?? checked - boolean state of the radio button.
- ?? defaultChecked - the HTML CHECKED attribute.
- ?? form – the form containing this element
- ?? name - the HTML NAME attribute.
- ?? type – the type of this form element
- ?? value - the HTML VALUE attribute.

Methods
- ?? blur()
- ?? click()
- ?? focus()

4.15 Reset

```
Interface Reset {
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void click();
  void focus();
};
```

The HTML Form Reset button.

Event Handlers
- ?? onBlur
- ?? onClick
- ?? onFocus

Properties
- ?? form – the Form containing this element
- ?? name – the name of this form element
- ?? type – the type of this form element
- ?? value - the HTML VALUE attribute.

Methods

- ?? blur()
- ?? click()
- ?? focus()

4.16 Select

```
Interface OptionSequence {
  Option item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface Select {
  readonly attribute Form form;
  readonly attribute long length;
  readonly attribute String name;
  readonly attribute OptionSequence options;
  attribute long selectedIndex;
  readonly attribute String type;
  void blur();
  void focus();
};
```

The HTML Form Select object.

Event Handlers

- ?? onBlur
- ?? onChange
- ?? onFocus

Properties

- ?? form – the Form that contains this element
- ?? length - number of options in the selection list.
- ?? name – the name of this form element
- ?? options[] - the array of Option objects, reflecting each of the options in the selection list in the order they appear.
- ?? selectedIndex – the index (position) of the currently selected option in the selection list.
- ?? type – the type of this form element

Methods

- ?? blur()
- ?? focus()

4.17 String

```
Interface String : ECMAString {
  String anchor(in String nameAttribute);
  String big();
  String blink();
  String bold();
  String fixed();
  String fontcolor(in String color);
  String fontsize(in long size);
  String italics();
```

```
  String link(in String hrefAttribute);
  String small();
  String strike();
  String sub();
  String sup();
};
```

This is an extension of the standard ECMAScript String object. The above IDL shows that this is derived from the ECMAScript object, but in reality it is an extension of the standard object.

Properties

- *<none>*

(Extra) Methods

- anchor() - returns the string, "<A NAME=nameAttribute">string</A>".
  - nameAttribute - value of the HTML NAME attribute.
- big() - returns the string, "<BIG>string</BIG>".
- blink() - returns the string, "<BLINK>string</BLINK>".
- bold() - returns thestring, "<B>string/B>".
- fixed() - returns the string, "<TT>string</TT>".
- fontColor() - returns the string, "<FONT COLOR=color>string</FONT>".
  - color - the value of the HTML COLOR attribute
- fontsize() - returns the string, "<FONTSIZE=size>string</FONTSIZE>".
  - size - the HTML FONTSIZE tag value.
- italics() - returns the string, "<I>string</I>".
- link() - returns the string, "<A HREF=hrefAttribute>string</A>".
  - hrefAttribute - the value of the HTML HREF attribute.
- small() - returns the string, "<SMALL>string</SMALL>".
- strike() - returns the string, "<STRIKE<string</STRIKE>".
- sub() - returns the string, "_{string}".
- sup() - returns the string, "^{string}".

4.18 Submit

```
Interface Submit {
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void click();
  void focus();
};
```

The HTML Form Submit button.

Event Handlers

- onBlur
- onClick
- onFocus

Properties

- form – the Form that contains this element

- ?? name – the name of this form element
- ?? type – the type of this form element
- ?? value - the HTML VALUE attribute.

Methods

- ?? blur()
- ?? click()
- ?? focus()

4.19 Text

```
Interface Text {
  readonly attribute String defaultValue;
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void focus();
  void select();
};
```

The HTML Form Text Object.

Event Handlers

- ?? onBlur
- ?? onChange
- ?? onFocus

Properties

- ?? defaultValue - the HTML VALUE attribute.
- ?? form – the Form that contains this element
- ?? name – the name of this form element
- ?? type - the HTML TYPE attribute.
- ?? value - the current string value of the field.

Methods

- ?? blur()
- ?? focus()
- ?? select()

4.20 TextArea

```
Interface TextArea {
  readonly attribute String defaultValue;
  readonly attribute Form form;
  readonly attribute String name;
  readonly attribute String type;
  attribute String value;
  void blur();
  void focus();
  void select();
```

```
};
```

The HTML Form TextArea object.

Event Handlers

- ?? onBlur
- ?? onChange
- ?? onFocus

Properties

- ?? defaultValue - the HTML VALUE attribute.
- ?? form – the From that contains this element
- ?? name – the name of this element
- ?? type - the HTML TYPE attribute.
- ?? value - the current value of the string field.

Methods

- ?? blur()
- ?? focus()
- ?? select()

4.21 Window


```
Interface WindowSequence {
  Window item(in unsigned long index);
  readonly attribute unsigned long length;
};
Interface Window {
  attribute String defaultStatus;
  readonly attribute Document document;
  readonly attribute WindowSequence frames;
  readonly attribute History history;
  readonly attribute long length;
  attribute Location location;
  readonly attribute String name;
  readonly Navigator navigator;

  readonly Window opener;
  readonly attribute Window parent;
  readonly attribute Window self;
  attribute String status;
  readonly attribute Window top;
  readonly attribute Window window;
  void alert(in String message);
  void clearTimeout(in long timeoutID);
  void close();
  boolean confirm(in String message);
  void open(in String url, in String name, in String features,
in boolean replace);
  String prompt(in String message);
  String prompt(in String message, in String inputDefault);
  String prompt(in String message, in long inputDefault);
  long setTimeout(in String expr, in long msec);
};
```

This provides basic window management functions. Note that only one window is supported, so there are restrictions on properties and methods that imply or act on multiple windows. Authors should be aware that browser implementations may restrict access to Window object information and navigation control under some circumstances, such as from scripts running within frames, in order to protect the privacy of user information, and maintain control by the host document/application.

Event Handlers

- ?? onLoad
- ?? onUnload

Properties

- ?? defaultStatus - the default string in the "status bar" (implementation dependent - see status below).
- ?? document - document object of the document in this window.
- ?? frames[] - an array of window objects that represent the frames in this window. Frames appear in the array in the order in which they appear in the HTML source code. If there are no frames, then this array is empty, and the length value below is zero.
- ?? history - the history object for URL's displayed in this window (note that this is global history list).
- ?? length - length of the frames array.
- ?? location - the location (URL) object for this window, representing the URL used to request the current document. Setting the value of window.location to an URL string sets the href value of the window's associated location object, and causes the window to navigate to the URL, if window.location is set to a new URL. Note that the requested URL in window.location may be different than the actual URL of the delivered document, which is stored in document.URL.
- ?? name - name of this window (implementation specific).
- ?? navigator – the pointer to the navigator object.
- ?? opener – the URL string of the document that opened the window.
- ?? parent - the parent window (of a frame).
- ?? self - pointer to this window object.
- ?? status - string to be displayed (perhaps transiently) in the status bar. This is implementation dependent, and cannot be counted on to be reliably displayed to the user. Its use is discouraged and is supported here for script compatibility.
- ?? top - topmost window in the window/frame hierarchy.
- ?? window - same as self.

Methods

- ?? alert() - display an "alert" message (implementation dependent). Note that this may not block in event handler code, so its use should be avoided in there.
  - ?? message - the message string to display.
- ?? clearTimeout() - clear the timer set by setTimeout.
  - ?? timeoutID - the ID of the timer.
- ?? close() – close (and cause to render) the window.
- ?? confirm() - display a "confirm" message (implementation dependent). Note that this may not block in event handler code, so its use should be avoided in there.
  - ?? Message - the message string to display.
- ?? open() – open a new window. The exact receiver behavior of this (single window or multiple window system, etc) is implementation dependent.
  - ?? url – the URL string of the document to open.
  - ?? name – string name of the window.
  - ?? features – string of features about the target window, of the form, "token[=value]". Any tokens are legal, but the ones a content author should expect to work are:
    - ?? height – height in pixels
    - ?? status – boolean to enable the status line
    - ?? width – width in pixels

- ?? replace – a boolean if true, replaces the current window
- ?? prompt() - display a "prompt" message (implementation dependent) and return the user entry. Note that this may not block in event handler code, so its use should be avoided in there.
  - ?? Message - the message string to display.
  - ?? inputDefault - the value first displayed in the prompt area.
- ?? setTimeout() - evaluate an ECMAScript expression after a period of time.
  - ?? expr - the expression to evaluate.
  - ?? msec - the milliseconds to wait before evaluation.

5 Event Handling

User interface events are handled in this model very differently than is proposed in DOM-2. These are done with a small set of methods to allow the simulation of events, and a small set of HTML attributes that are "handler-like". The methods allow programmatic control over causing certain events, and the callback "handlers" are "evoked" (more on the quotes below) when the event occurs. There are no object attributes that represent these “handlers”. The ECMA Script code for these can only be defined by the HTML tags.

5.1 Methods

The methods are:

- ?? blur()
- ?? click()
- ?? focus()
- ?? select()

blur()

This removes focus from the object.

click()

This simulates a mouse click on the object.

focus()

This sets the focus on the object.

select()

This selects in a field in some input objects as if the user had used the mouse to highlight and select an item.

5.2 Handlers

The callback "handlers" are actually string values containing in line simple or compound ECMAScript statements, and not actual functions. So these are never actually invoked, but are documented here as "functions" for conceptual clarity only.

5.2.1 Scope

The scope chain for executing an event handler begins with the object which emits the event and proceeds upwards through the object reference hierarchy as follows: {Link,Input*}, [Form], Document, Window (where Input* refers to any of the Form input element types).

Within the context of a global function, i.e., a function declared in a <script> element, 'this' is bound to the applicable Window object. The applicable Window object is determined by the scope chain of the caller of such a function.

5.2.2 Handler Context

Within the context of an event handler, 'this' is bound to the object which emits the event.

5.2.3 Return Values

The return value is that set using an ECMAScript 'return' statement in the event handler compound statement.

5.2.4 Handler Summary

The handler "functions" are:

- ?? onBlur()
- ?? onChange()
- ?? onClick()
- ?? onFocus()
- ?? onLoad()
- ?? onMouseOut()
- ?? onMouseOver()
- ?? onSubmit()
- ?? onUnLoad()

onBlur

This HTML attribute allows the user to set an ECMAScript compound statement to execute when focus is lost on the object. "this" is any Input object as appropriate. Any return value is ignored.

onChange

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the field value is changed by the user, and can be used to validate fields before submission. "this" is one of {Select, Text, TextArea} as appropriate. Any return value is ignored.

onClick

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the user clicks on the object. "this" is one of {Button, Checkbox, Link, Radio} as appropriate. A return value of 'false' will cancel the default action, and 'true' will perform the default action.

onFocus

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the object gets focus. "this" is any Input object as appropriate. Any return value is ignored.

onLoad

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the object (window, frame, or image) is fully loaded. "this" is the Window or Image object as appropriate. Any return value is ignored.

onMouseOut

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the mouse is moved out of the object's defined area. "this" is the Link object. Any return value is ignored.

onMouseOver

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the mouse is moved into the object's defined area. "this" is the Link object. A return value of 'true' will prevent the system from displaying the URL in the status bar.

onSubmit

This HTML attribute allows the user to set an ECMAScript compound statement to execute when the user submits a form. "this" is the Form object. A return value of false will prevent the submission.

onUnload

This HTML attribute allows the user to set an ECMAScript compound statement to execute just prior to when the document in this window is unloaded. Any return value is ignored.

6 Exception Handling

This version of this document only contemplates an ECMAScript, edition 2 binding. There is no exception handling in ECMAScript, edition 2. Therefore, exception handling, if any, is entirely implementation dependent.

7 Security

ECMAScript in a frame is allowed to access methods and properties in other frames whose URLs specify the same host name and port. Note that there may be security policies that restrict usage of the methods, but that is implementation dependent.

8 Standard Color Names

The supported color names are those defined in CSS1. The values for the colors shall be from the HTML 4 specification.

9 ECMA Script Language Binding (Normative)

Note that this is subordinate to the IDL definitions above. If there is any inconsistency, then refer to the IDL.

```
Object Anchor
  Properties:
    String name


Object Button
  Properties:
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void click()


Object Checkbox
  Properties:
    Boolean checked
    Boolean defaultChecked (readonly)
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void click()
    void focus()


Object Document
  Properties:
    String alinkColor
    Array anchors (readonly of type Anchor)
    String bgColor
    String cookie
    String fgColor
    Array forms (readonly of type Form)
    Array images (readonly of type Image)
    String lastModified
    String linkColor
    Array links (readonly of type Link)
    String location (readonly)
    String referrer
    String title (readonly)
    String URL (readonly)
    String vlinkColor
  Methods:
    Void clear()
    void close()
    void open(String mimeType)
    void write(String expr1, String expr2, ...)
    void writeln(String expr1, String expr2, ...)
```

```
Object Form
  Properties:
    String action
    Array elements (readonly of type Object)
    String encoding
    Number length (readonly)
    String method
    String name
    String target
  Methods:
    void submit()


Object Hidden
  Properties:
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value


Object History
  Methods:
    void back()
    void forward()
    void go(Number delta)
    void go(String location)

Object Image
  Properties:
    Number border
    Boolean complete (readonly)
    Number height
    Number hspace
    String lowsrc
    String name
    String src
    Number vspace
    Number width


Object Link
  Properties:
    String hash
    String host
    String hostname
    String href
    String pathname
    String port
    String protocol
    String search
    String target


Object Location
```

```
  Properties:
    String hash
    String host
    String hostname
    String href
    String pathname
    String port
    String protocol
    String search


Object Navigator
  Properties:
    String appCodeName (readonly)
    String appName (readonly)
    String appVersion (readonly)
    String userAgent (readonly)


Object Option
  Properties:
    Boolean defaultSelected (readonly)
    Number index (readonly)
    Boolean selected
    String text (readonly)
    String value


Object Password
  Properties:
    String defaultValue (readonly)
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void focus()
    void select()


Object Radio
  Properties:
    Boolean checked
    Boolean defaultChecked (readonly)
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void click()
    void focus()


Object Reset
```

```
  Properties:
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void click()
    void focus()


Object Select
  Properties:
    Form form (readonly)
    Number length (readonly)
    String name (read only)
    Array options (readonly of type Option)
    Number selectedIndex
    String type (readonly)
  Methods:
    void blur()
    void focus()


Object String (extensions to standard object - not derived)
  Methods:
    String anchor(String nameAttribute);
    String big();
    String blink();
    String bold();
    String fixed();
    String fontcolor(String color);
    String fontsize(Number size);
    String italics();
    String link(String hrefAttribute);
    String small();
    String strike();
    String sub();
    String sup();


Object Submit
  Properties:
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void click()
    void focus()


Object Text
  Properties:
    String defaultValue (readonly)
```

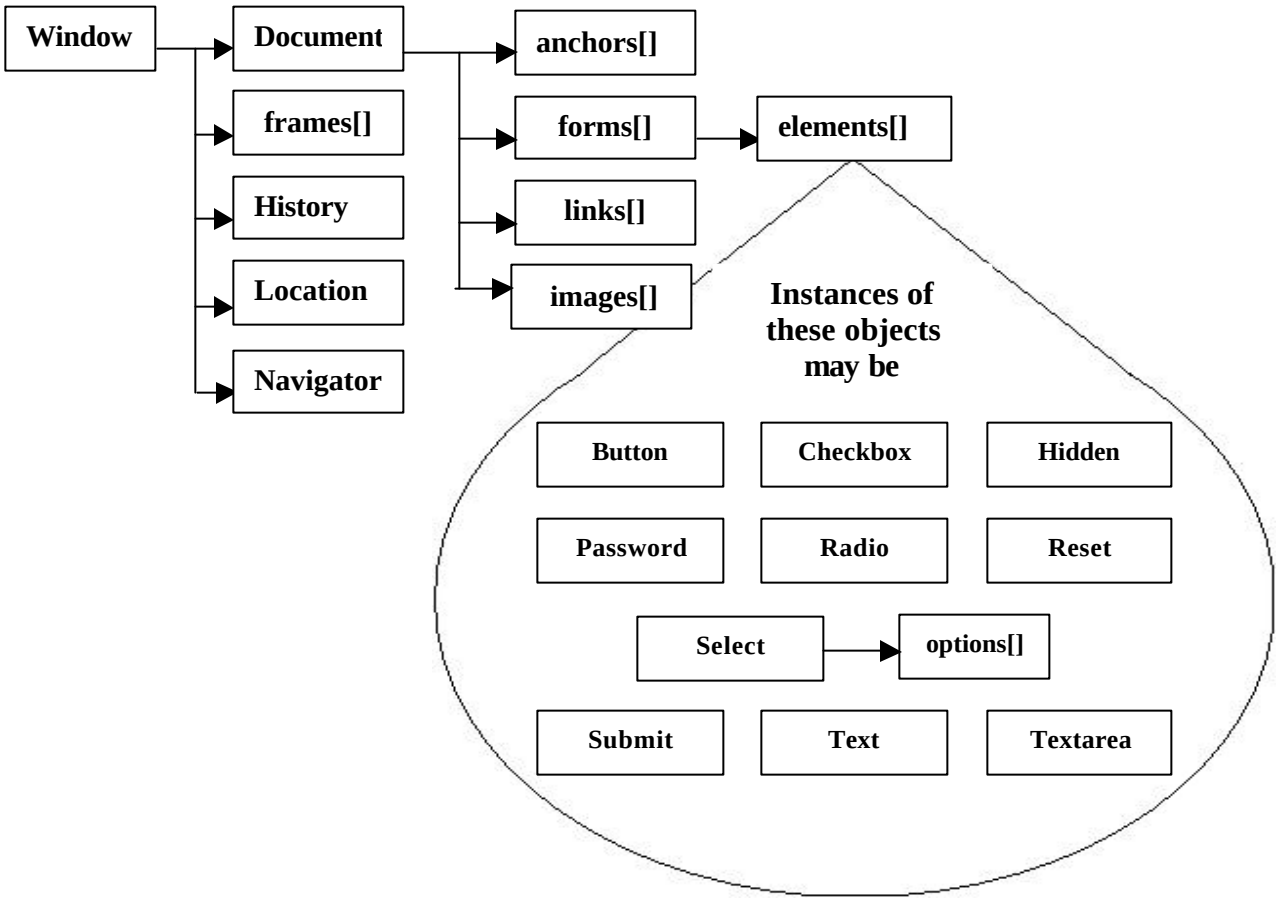
```
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void focus()
    void select()


Object TextArea
  Properties:
    String defaultValue (readonly)
    Form form (readonly)
    String name (readonly)
    String type (readonly)
    String value
  Methods:
    void blur()
    void focus()
    void select()


Object Window
  Properties:
    String defaultStatus
    Document document (readonly)
    Array frames (readonly of type window)
    History history (readonly)
    Number length (readonly)
    Location location
    String name (readonly)
    Navigator navigator (readonly)
    Window opener (readonly)
    Window parent (readonly)
    Window self (readonly)
    String status
    Window top (readonly)
    Window window (readonly)
  Methods:
    void alert(String message)
    void clearTimeout(Number timeoutID)
    void close()
    Boolean confirm(String message)
    open(String url, String name, String features, boolean replace)
    String prompt(String message, Number inputDefault)
    Number setTimeout(String expr, Number msec)
```

10 Annex A – Object Relationships (Informative)

Each of the objects in section 4 is defined in following sections by means of an IDL Interface specification, one Interface specification per object. No object has an inheritance relationship with any other object. However, some of the objects, e.g., Window, Document, Select, have attributes that reference other object instances or sequences of other object instances. The following diagram shows how object instances are linked as a result of these references. The arrow means "has attribute which references." The notation "<object-name>[]" is a reference to a object which is a sequence of <object-name>:



Draft SMPTE Standard SMPTE XXXX **The Local Identifier (lid:) URI Scheme**	Draft i 20-April- 2001

1 Scope

This document defines the "lid:" Uniform Resource Identifier (URI), and describes how it is used to identify instances of resources, such as web pages and graphics files, that are transmitted through uni-directional means, such as a television broadcast.

2 References and Organization

2.1 Normative References

The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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.

[URI] Berners-Lee, T., Masinter, L. and R. Fielding, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[UUID] ISO/IEC 11578:2000, Information technology, "Open Systems Interconnection – Remote Procedure Call", Annex A, "Universal Unique Identifier"

2.2 Table of contents

3 Introduction

Content resources delivered by a one-way broadcast must be identified, stored in a storage system used by the receiver as they are received, and referenced by a uniform scheme for access by applications and systems. Broadcast receivers may use different types of storage devices, therefore content broadcasters and application developers need a standard syntax for resource storage and reference that does not depend on the specific device or directory syntax, such as the file: URI scheme. A lid: can be bound to a resource entity during authoring and distribution, and may be used to name a device-independent storage location for the entity. The lid: scheme is syntactically similar to the http: scheme, but it is not intended to resolve lid: identifiers to locations outside the broadcast stream or local storage system, as is the case for http: DNS and resource resolution.

The "lid:" URI scheme enables content creators to assign an "authority" value that is globally unique. The lid: scheme supports relative paths for resource retrieval, so the authority component can be separately identified in applications to allow relative path references similar to http: and other URI references.

A single lid: can be used to identify different resource instances over time, and will resolve in a receiver to the last instance received with an equivalent lid:. Appropriate lid: identifiers can reduce the storage of redundant instances of resources for better memory efficiency. Storage management for lid: entities is implementation specific and beyond the scope of this specification, but it can be assumed that memory will be finite, and so will the period of persistence of any lid: entity. Applications using lid: should be designed to handle the case where resources have been deleted over time due to storage limitations.

4 Definition of the Local Identifier ("lid:") URI Scheme

The `"lid:"` URI scheme describes URI references consisting of a sequence of characters, which are independent of their coding in octets in any particular character set. The lid: URI fully complies with the "Uniform Resource Identifiers (URI): Generic Syntax" RFC 2396 [URI] except for the overloading of the authority field in the deprecated form. [URI].

The layout of the `"lid:"` URI follows the "generic URI" syntax:

lid://<userinfo>@<host>:<port><path>?<query>#<fragment>

"**userinfo**" is an optional string that enables message ID syntax forms of the authority field and, in combination with the "host" field, complies with the mid: scheme syntax defined in [MID].

"**host**"is a string whose root is a registered domain name or a uuid [UUID] in string form. Note that the uuid form is deprecated and is intended to support past common practice.

"**port**" is a string to allow syntactic compatibility with [HTTP] and has no semantic meaning.

"**path**" is a slash-separated string of components identical to the http: scheme syntax as defined in [HTTP].

"**query**" and "**fragment**" are content type dependent strings compliant with [URI].

Relative path syntax, as described in section 3 of reference [URI] is also permitted syntactically, but must only be used in cases where there is a guaranteed mechanism to resolve the absolute path (i.e. the BASE URI is well-defined). Practical delivery considerations may require that lid: identified resources be delivered on broadcast channels using absolute paths to enable real-time storage in sequence of resource arrival, but relative path resolution must be supported for lid: resource retrieval, assuming an application specifies the base of the URI by other means.

The following are examples of lid:'s:

lid://xbc.com/EveningNews/11-March-01/Pacific/main.html

lid://4F4182C71C1FDD4BA0937A7EB7B8B4C1@mail.xbc.com

A deprecated form of usage is to permit "host" to be an encoded UUID [UUID]. While technically, the UUID name space overlaps the domain name space, in practice, a collision is entirely improbable. Examples of this deprecated form using UUID is:

lid://4F4182C71C1FDD4BA0937A7EB7B8B4C1/images/logo.gif

lid://4F4182C7-1C1F-DD4B-A093-7A7EB7B8B4C1/images/logo.gif

The UUID is represented as an ASCII hex encoding resulting in 32 characters. Note that two syntaxes are permitted – one with some specifically placed separating hyphens, and one without (see the BNF definition below).

When different resources are received with matching lid:'s, the most recently received resource should be referenced using that lid:. Thus a lid: may refer to different resources over time.One lid: URI can be assigned for all instances of a resource, or multiple unique URI's can be assigned, one for each instance of the resource.

5 Resolution Rules

A "lid:" URI is used to label a resource. Certain parts of the URI are ignored for the purposes of comparison, when the lid: is used for retrieval, or to replace a previously transmitted resource with an equivalent URI. When testing for equivalence, the query and fragment identifiers (i.e. characters in the lid: including and following the first "?" or "#" character) in a lid: URI are

ignored. Notwithstanding, references using fragment and query identifiers may function in ways defined by the content type being referenced..

When comparing two URIs to decide if they match or not, a receiver should use a case-sensitive octet-by-octet comparison of the entire URIs, with these exceptions:

- ?? a port that is empty or not given is equivalent to the default port for http:, which is 80;
- ?? comparisons of host names shall be case-insensitive;
- ?? comparisons of scheme names shall be case-insensitive;
- ?? an empty abs_path is equivalent to an abs_path of "/"

Characters other than those in the "reserved" and "unsafe" sets (see [URI]) are equivalent to their ""%" HEX HEX" encoding. For example, the following three URIs are equivalent:

lid://abc.com:80/~smith/home.html

lid://ABC.com/%7Esmith/home.html

lid://ABC.com:/%7esmith/home.html

Unlike http:, a lid: URI is not locatable without more information and is thus, URN-like in the generic definition in that a URN is associated to a resource and independent of the resource's location. Therefore, the details of resolution of the location of a lid: is application dependent.

6 Normalization and Equivalence

In many cases, different URI strings may actually identify the same resource. For example, the host names used in the URL are case insensitive, so the URL <lid://www.XBC.com> is equivalent to <lid://www.xbc.com>. In general, the rules for equivalence and definition of a normal form, if any, are scheme dependent. When a scheme uses elements of the common syntax, it will also use the common syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL with an explicit ":port", where the port is the default for the scheme, is equivalent to one where the port is elided.

7 Local Identifier Syntax BNF

The collected BNF for "lid:" URI's is as follows:

```
lid           = "lid" ":" "//" authority [ abs_path ] [ "?"
query ] [ "#" fragment ]
authority     = server | uuid
server        = as defined in RFC2396
abs_path      = as defined in RFC2396
query         = as defined in RFC2396
fragment      = as defined in RFC2396
uuid          = uuid_simple | uuid_idl
uuid_simple   = 32hex
uuid_idl      = 8hex "-" 4hex "-" 4hex "-" 4hex "-" 12hex
hex           = as defined in RFC2396

Note 1: the notation <n>(element) means exactly <n>
occurrences of (element); e.g., 32hex means exactly 32 hex
digits.

Note 2 that the use of the uuid authority element is
deprecated.
```

8 Security Considerations

The local identifier URI scheme is subject to the same security implications as in general URI [URI] schemes, so the usual precautions apply. This means that some local identifier URI's may refer to resources that are not available (because they have not been received, for example), or to resources that have been received, but were intentionally misidentified. The security issues associated with this mislabeling, as well as the security issues associated with the use of HTML content which is broadcast are the same as those identified in section 11.1 of RFC 2387[MIME].

Appropriate security mechanisms should be used in the delivery of content identified by lid: URI's. These include protection of the broadcast signal by data encryption and conditional access methods, and protection of content prior to broadcast so that invalid lid:'s are not created, and valid lid:'s are not modified.

9 Bibliography

[HTTP] Fielding, R. et al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616,

[MID] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2111

[MIME] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387

[RFC2718] L. Masinter, et al, "Guidelines for new URL Schemes", RFC 2718, November 1999.

[BCP35] R. Petke, I. King, "Registration Procedures for URL Scheme Names", BCP 35 and RFC 2717, November 1999.

Appendix A (Informative)

Converting Other URI Schemes to lid:

URL references using schemes such as http:, ftp:, and file: can be converted to valid lid:'s by changing the scheme component, and using the original name, host, path, fragment, and query components. This is useful, for instance, to deliver resources stored on Internet servers over a broadcast channel. Note that the original URL "port" and "password" fields have no semantic definition in lid:.

Example:

An Internet resident resource at the location:

http://www.xbc.com/tv/text.txt

Could be packaged in a broadcast stream with a header containing the resource identifier;

lid: //www.xbc.com/tv/text.txt

And that resource identifier could be used to store the text.txt entity in memory with a derived directory entry, which would be matched by the following lid: reference in an HTML document;

href=lid://www.xbc.com/tv/text.txt?ID="myProgram"

Proposed SMPTE Standard SMPTE 357M **Declarative Data Essence, IP Multicast Encapsulation**	

1 Scope

This document defines a standard for the encapsulation of declarative data essence using IP Multicast. This is done in a transport-independent manner and relies solely on standard IP Multicast techniques.

2 References and Organization

2.1 Normative References

The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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.

SMPTE Proposed Standard 363M, "Declarative Data Essence, Content Level 1"

IETF RFC 2327, "SDP: Session Description Protocol"

IETF RFC 2974, "Session Announcement Protocol" [SAP]

SMPTE Proposed Standard 364M, "The Unidirectional Hypertext Transfer Protocol" [UHTTP]

ISO/IEC 11578:2000, Information technology, "Open Systems Interconnection – Remote Procedure Call", Annex A, "Universal Unique Identifier" [UUID]

IETF RFC 1112, "Host Extensions for IP Multicasting

IETF RFC 768, "User Datagram Protocol" [UDP]

IETF RFC 791, "INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION" [IP]

2.2 Document Organization

3 Introduction

This specification defines the transmission of declarative content across terrestrial (over the air), cable, and satellite systems as well as over the Internet. In addition, it will also bridge between networks - for example data on an analog terrestrial broadcast must easily bridge to a digital cable system. This design goal was achieved through the definition of a transport-independent content format and the use of IP. Since IP bindings already exist for each of these video systems, we can take advantage of this work.

IP multicast is the mechanism for broadcast data delivery. Content creators should assume IP addresses may be changed downstream, and therefore should not use them in their content. The transport operator is only responsible for making sure that an IP address is valid on the physical network where they broadcast it (not for any re-broadcasting). When possible, content creators should use valid IP multicast addresses to minimize the chance of collisions. Some systems may have two-way Internet connections. Capabilities in those systems are outside the scope of this document and are described by the appropriate Internet standards.

Transport operators should use the standard IP transmission system for the appropriate medium (IETF, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver automatically locates and delivers IP datagrams associated with the TV broadcast. The mechanism for tuning video and connecting to the appropriate data stream is implementation and delivery standard specific and is not specified in this framework.

4 Announcement Protocol

Announcements are used to announce currently available programming to the receiver. The IP multicast addresses and ports for resource transfer and for triggers are announced using SDP announcements (RFC 2327). The SDP Header is preceded by an 8-byte SAP header. Announcements are sent on a well-known address (224.0.1.113) and port (2670). This address and port have been registered with the IANA.

`v=0`	SDP Version, required to be 0.
`o=username` *`sid version`* `IN IP4` *`ipaddress`*	Owner & session identifier, defined as usual in SDP spec. Username is "-", network type is IN, address type is IP4. Sid identifies an announcement for a particular broadcast (it can be a permanent announcement for all programming on a broadcast channel or for a particular show). Version indicates the version of the message. These values allow receivers to match a message to a previous message and know whether it has changed. Session ID and Version should be NTP values as recommended in SDP.
`s=name`	Session name, required as in SDP spec.
`i=, u=`	Optional, as in SDP spec.
`e=, p=`	E-mail address or phone number, at least one required in SDP spec.
`b=CT:`*`number`*	Optional in SDP spec, but Required here. Bandwidth in kbps as in the SDP spec. Bandwidth of the broadcast data can be used by receivers to choose among multiple versions of enhancement data according to the bandwidth the receiver can handle.
`t=`*`start stop`*	As in SDP spec gives start and stop time in NTP format. With programs stored on tape, at times it will not be possible to insert new announcements, so start times on tape could be incorrect. In this case, the start time should be set to the original broadcast time and the stop time set to 0. This is the standard for an unbounded session. Assumptions are then made about the stop time (see RFC 2327). A new announcement with the same sid and different version for the same broadcast station replaces the previous one. It is preferred that a tool read the tape and generate announcements with correct start and stop times, but not required. Content creators can choose to use only a station ID and not provide information about individual programs.
`a=UUID:`*`UUID`*	Optional. The UUID should uniquely identify the enhancement (for example, a different UUID for each program), and can be accessed using the trigger receiver object. In analog TV and many types of digital TV broadcast data is tied tightly to A/V. Each virtual channel has its own private network associated with it. In other systems, enhancements for many virtual channels can be carried on the same network. These systems can use the UUID to link a TV broadcast with a particular enhancement. How that association is indicated is beyond the scope of this document. One technique would be to place the UUID in electronic program guide information. Use ASCII HEX to encode UUIDs.
`a=type:tve`	Required. Indicates to the receiver that the announcement refers to an enhancement related to this specification.
`a=`*`lang`*`,` `a=`*`sdplang`*	Optional, as in SDP spec.
`a=tve-type:<types>`	Optional. `tve-type:` specifies an extensible list of types that describe the nature of the enhancement. It is a session-level attribute and is not dependent

	on charset. `a=tve-type:primary` Optional. `tve-type:primary` specifies that this will be the primary enhancement stream associated with the currently playing video program whenever this enhancement's trigger stream is active. If `tve-type:primary` is not specified, the TVE stream is never the primary enhancement stream associated with video. This, like all `tve-type:` attributes, is a session level attribute. This attribute can be used by receivers to implement automatic loading of primary video enhancement streams. The actual display of and switching between enhancement streams is handled by the trigger streams.
`a=tve-size:`*`Kbytes`*	Required. `tve-size:` provides an estimate of the high-water mark of cache storage in kilobytes that will be required during the playing of the enhancement. This is necessary so that receivers can adequately judge whether or not they can successfully play an enhancement from beginning to end.
`a=tve-level:`*`x`*	Content level identifier, where x is 1.0 for this version of the framework (optional, default is 1.0).
`a=tve-ends:`*`seconds`*	Optional, specifies an end time relative to the reception time of the SDP announcement. *Seconds* is the number of seconds in the future that this announcement is valid. *Seconds* may change (count down) as an announced session progresses. This attribute, when present, overrides the default assumptions for end times in unbounded announcements.
`m=data` *`portvalue`*`/2 tve-file/tve-trigger c=IN IP4` *`ipaddress/ttl`*	As in SDP spec. Compact form specifying 2 ports on same address

When there are multiple alternative enhancement streams for the same video program, they must all be announced at the media level of the same SDP announcement. All enhancement streams announced in the same SDP announcement are considered to be mutually exclusive variants of the primary enhancement stream. The receiver can choose between them based on media level attributes. For example, the `a=lang` field can be used at the media level to choose between language variants of the primary enhancement.

Each media section for the tve-file media type begins the next enhancement definition.

A longer form is available if the content creator or transport operator wants to use different IP addresses and ports for the data stream and trigger stream:

`m=data` *`portvalue`* `tve-file c=IN IP4` *`ipaddress/ttl`*	Alternative form for specifying addresses and ports (for file protocol, as in SDP spec)

m=data *portvalue* tve-trigger c=IN IP4 *ipaddress/ttl*	For control protocol, as in SDP spec.

Announcement Example:

```
v=0
o=- 2890844526 2890842807 IN IP4 tve.niceBroadcaster.com
s=Day & Night & Day Again
i=A very long TV Soap Opera
e=help@niceBroadcaster.com
a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
a=type:tve
a=tve-level:1.0
t=2873397496 0
a=tve-ends:30000
a=tve-type:primary
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.1.112/127
b=CT:100
a=tve-size:1024
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.0.1/127
b=CT:1024
a=tve-size:4096
```


5 Trigger Protocol

The trigger protocol carries a single trigger in a single UDP/IP multicast packet. Triggers are real-time events broadcast inside IP multicast packets delivered on the address and port defined in the SDP announcement for the enhanced TV program (see Announcements). The trigger protocol is thus very lightweight in order to provide quick synchronization.

6 Resource Transfer: UHTTP

A one-way IP multicast based resource transfer protocol, the Unidirectional Hypertext Transfer Protocol (UHTTP) is defined in IETF RFC xxxx. UHTTP is a simple, robust, one-way resource transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This resource transfer protocol is appropriate for IP multicast over television vertical blanking interval (IPVBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems.

Web pages and their related resources (such as images and scripts) are broadcast over UDP/IP multicast in the related TV signal. An announcement broadcast by the TV station tells the receiver which IP multicast address and port to listen to for the data. The only data broadcast to this address and port are resources intended for display as Web content.

While HTTP headers preceding resource content are optional in the UHTTP protocol, they are required when the protocol is used for DDE. Compliant receivers must support “gzip” content encodings as specified by the "Content-Encoding" HTTP header field.

DRAFT

7 Appendix A: Using Enhanced TV (Informative)

Television enhancements are comprised of three related data sources: announcements (delivered via UDP using SAP/SDP), content (delivered via UHTTP), and triggers (delivered via the trigger protocol over UDP).
Announcements are broadcast on a single well-known multicast address and have a time period for which they are valid. This time period is expressed via the `"t="` and `"a=tve-ends:"` lines within the SDP record.
Announcements also indicate the multicast address and port number that the client can listen in on to receive the content and triggers.
The announcement also contains information that the client can optionally use to help decide whether to automatically start receiving triggers and content information. This may include `a=tve-type`, `lang=`, and `keywds=` attributes that provide additional information to the client about the announced enhancements. For example, announcements with an optional `a=tve-type:primary` attribute may be used by the client to implement an "auto-play" feature. Multiple `a=tve-type` attributes may appear in a given announcement and are not mutually exclusive.

8 Appendix B: Example Broadcast (Informative)

The following is a simple example of a television enhancement, delivered via transport type B (multicast IP). The example consists of three parts: the announcement (announced via SDP/SAP), the content (delivered via UHTTP), and the triggers (delivered in UDP packets).

The experience consists of a screen with a 60% sized embedded live TV object, with some text below it. During the show, a trigger may arrive that will cause an image of the word "MURDER" to appear below the text. If the user chooses to click on the TV object, they will be returned to full screen video, and away from the enhanced experience.

Announcement:

The following announcement packet is sent via UDP to the multicast IP address: 224.0.1.113, port: 2670.

The announcement consists of an 8 byte SAP header followed by an SDP text payload.

The values for the SAP header fields for the announcement::

Field Name (size)	**Value**	**Description**
Version (3 bits)	1	SAP version
Message Type (3 bits)	0	Session description announcement packet
Encrypted (1 bit)	0	Not encrypted
Compressed (1 bit)	0	Not compressed
Authentication Length (1 byte)	0x00	No authentication
Message ID Hash (2 bytes)	0x3464	Hash of payload text
Originating Source Address (4 bytes)	209.240.195.6	IP address of originating host

Complete SAP header would be eight bytes: 0x20, 0x00, 0x34, 0x64, 0xd1, 0xf0, 0xc3, 0x06

The remaining bytes in the announcement packet would contain the following text payload:

```
v=0
o=- 2890844526 2890842807 IN IP4 tve.niceBroadcaster.com
s=Day & Night & Day Again
i=A very long TV Soap Opera
e=help@niceBroadcaster.com
a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
a=type:tve
a=tve-level:1.0
a=tve-ends:1800
a=tve-type:primary
t=2873397496 0
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.1.112/127
b=CT:40
a=tve-size:1024
```

These fields indicate the following:

`v=0`	SDP version zero
`o=- 2890844526 2890842807 IN IP4 tve.niceBroadcaster.com`	Originating host information
`s=Day & Night & Day Again`	Session name
`i=n very long TV Soap Opera`	Session description
`e=help@niceBroadcaster.com`	Contact information about the session
`a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`	Unique identifier (UUID) for the session
`a=type:tve`	This is a television enhancement
`a=tve-level:1.0`	Content level 1
`a=tve-ends:1800`	Session ends 30 minutes from now
`a=tve-type:primary`	This session is the primary enhancement to the video
`t=2873397496 0`	Session began at a particular time
`m=data` *`52127`*`/2 tve-file/tve-trigger`	File and trigger data is available on ports *52127* and *52127+1*
`c=IN IP4` *`224.0.1.112`*`/127`	Data will be broadcast on multicast address *224.0.1.112*
`b=CT:40`	This session will have a maximum bandwidth of 40kbps
`a=tve-size:3`	This session will require a maximum amount of caching of 3k bytes

Content:

The content data for the enhancement is delivered via UHTTP packets transmitted (as specified by the announcement) to multicast address *224.0.1.112*, to port *52127*.

This content would consist of three original source files, two HTML documents and a PNG image. The experience consists of a screen with a 60% sized embedded live TV object, with some text below it. During the show, a trigger may arrive that will cause an image of the word "MURDER" to appear below the text. If the user chooses to click on the TV object, they will be returned to full screen video, and away from the enhanced experience

The first would be referred to by the URL:

`<lid://nicebroadcaster.com/show27/launch.html>`, and consists of the following text:

```
<HTML>

<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>
```

```
<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>


<BODY bgcolor="magenta">


<A href="tv:">
<OBJECT data="tv:" width="60%" height="60%" align="center">
</OBJECT>
</A>


<BR>
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>

<IMG name="sceneimage" align="center" src="">

</BODY>
</HTML>
```

The other files consist of a 2nd HTML file and a PNG image, containing an image of the word "MURDER" in big red letters. It's URL will be specified as `<lid://nicebroadcaster.com/show27/murder.png>`

These files are combined together into a single multipart MIME entity, which will make up the full payload of the UHTTP transmission.

```
Content-Base: lid://nicebroadcaster.com/show27
Content-Length: 2264
Content-Type: Multipart/Related; boundary=example98203804805

--example98203804805
Content-Location: launch.html
Content-Length: 450
Content-Type: text/html

<HTML>

<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>


<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>


<BODY bgcolor="magenta">


<A href="tv:">
<OBJECT data="tv:" width="60%" height="60%" align="center">
</OBJECT>
</A>


<BR>
```

```
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>

</BODY>
</HTML>
--example98203804805
Content-Location: murder.html
Content-Length: 475
Content-Type: text/html


<HTML>
<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>

<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>

<BODY bgcolor="magenta">

<A href="tv:">
<OBJECT data="tv:" width="60%" height="60%" align="center">
</OBJECT>
</A>

<BR>
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>

<IMG align="center" src="murder.png">

</BODY>
</HTML>
--example98203804805
Content-Location: murder.png
Content-Length: 1289
Content-Type: image/png
```


binary resource data for murder.png image

```
--example98203804805
```

This data multipart entity, (of total length, including headers of 2400 bytes) would be transmitted via UHTTP, in three packets. The first two packets would contain the original data (each containing 1200 bytes of original data as payload) and the third containing the exclusive-or of the first and second payloads as forward error correction data.

The UHTTP headers for each of the three packets would be as follows:

Field (size)	Packet 1 value	Packet 2	Packet 3	Description

		value	value	
Version (5 bits)	00000	00000	00000	UHTTP version
ExtensionHeader (1 bit)	0	0	0	No extension headers in this example
HTTPHeadersPrecede (1 bit)	1	1	1	HTTP headers precede data
CRCFollows (1 bit)	0	0	0	No CRC Follows
PacketsInXORBlock (1 byte)	3	3	3	Number of packets in each XOR FEC block
Retransmit Expiration (2 bytes)	1800	1800	1800	This will be retransmitted for the next 1800 seconds (this value will decrease as the show progresses)
TransferID (16 bytes)	0x1432 3ab4123 ab4567c d89ef05 67cd89e f0	same	same	UUID for this transmission
Resource Size (4 bytes)	2400	2400	2400	Size of payload
SegStartByte (4 bytes)	0	1200	2400	Offset into the stream where this packet's payload starts

In this example, the 28 UHTTP header bytes for the first packet would be:

```
0x02, (version, options)
0x03, (packets in XOR block)
0x07, 0x08, (retransmit expiration)
0x14, 0x32, 0x3a, 0xb4, 0x12, 0x3a, 0xb4, 0x56, 0x7c, 0xd8, 0x9e, 0xf0, 0x56,
0x7c, 0xd8, 0x9e, 0xf0, (Resource ID)
0x00, 0x00, 0x09, 0x2e, (Resource Size)
0x00, 0x00, 0x00 (Segment Start Byte Offset)
```

Following the header in the first packet, would be the first 1200 bytes of the MIME-encoded payload. Following the header in the second packet would be the last 1175 bytes of the MIME-encoded payload. Following the UHTTP header in the third packet would be 1200 bytes, where each byte was the exclusive-or of the corresponding byte offsets in the first two packets.

These packets would then be transmitted repeatedly during the session. The header values for each packet would remain the same, with the exception of the Retransmit Expiration field. The value of this field would decrease as the end of the transmission of the UHTTP packets drew near.

Triggers:

The following trigger would be sent after the data was first transmitted to trigger the beginning of the enhanced television experience:

```
<lid://nicebroadcaster.com/show27/launch.html>[name:Day & Night & Day Again Interactive]
```

This trigger content would be encapsulated in a UDP packet and sent to multicast address: *224.0.1.112*, port *52127+1* (as specified by the announcement)

This trigger packet would also be transmitted periodically later on, to allow viewers who tune in late to join in the fun.

Later on, during the program, the content creator might send the following trigger to the same multicast address and port to make the content change to reflect the fact that a murder scene has just begun in the program:

```
<lid://nicebroadcaster.com/show27/launch.html>[s:window.top.location.href="lid://nicebroadcaster.com/show27/murder.html"]
```

This trigger would cause the active enhancement page (if it matched the URL in the trigger) to load the new page, "murder.html", which would cause the murder.png image to be displayed within the page. If the specified URL was not currently being displayed, the trigger would be ignored because this trigger does not include a [name:] attribute.

Near the end of their program, they might send the following trigger to tell their interactive application to shut down. This would allow them to more accurately synchronize with the end of the program, rather than relying on the session timing information in the announcement.


```
<lid://nicebroadcaster.com/show27/launch.html>[script:window.location="tv:"]
```

9 Appendix C: Glossary

Announcements: Announcements are used to announce currently available programming to the receiver.
Binding: In the context of this document, abinding is the definition of how the transport specifications are encoded on a specific video network standard.
Content creator: In the context of this document, a content creator has the role of originating the content components of the television enhancement including graphics, layout, interaction, and triggers.
Datagram: a block of data that is "smart" enough (actually, which carries enough information) to travel from one Internet site to another without having to rely on earlier exchanges between the source and destination computer.
Essence: raw programme material.
FEC (Forward Error Correction)
FTP (File Transfer Protocol): A standard for finding and transferring files on the Internet.
HTTP (Hypertext Transfer Protocol): a set of instructions for communication between a server and a World Wide Web client.
IANA (Internet Assigned Numbers Authority): the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, and options, codes and types. The currently assigned values are listed in the Assigned Numbers document.
IETF (Internet Engineering Task Force): the IETF is a large, open community of network designers, operators, vendors, and researchers whose purpose is to coordinate the operation, management and evolution of the Internet, and to resolve short-range and midrange protocol and architectural issues. It is a major source of proposals for protocol standards, which are submitted to the IAB for final approval. The IETF meets three times a year and extensive minutes are included in the IETF Proceedings.
IP (Internet Protocol): This protocol is one of the languages computers connected to the Internet use to communicate.
IP multicast: A one-to-many transmission, in contrast to Unicast/Broadcast. An extension to the standard IP network-level protocol. RFC 1112, Host Extensions for IP multicasting, authored by Steve Deering in 1989, laid the groundwork for IP multicasting. The RFC describes IP multicasting as: "the transmission of an IP datagram to a 'host group', a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same 'best-efforts' reliability as regular unicast IP datagrams. The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time."
ISO (International Organization for Standardization): a voluntary, non treaty organization founded in 1946 which is responsible for creating international standards in many areas, including computers and communications. Its members are the national standards organizations of the 89 member countries, including ANSI for the U.S.
MIME (multipart/signed, multipart/encrypted content-types) a protocol for allowing e-mail messages to contain various types of media (text, audio, video, images, etc.).
NABTS (North American Basic Teletext Specification).
Receiver: In the context of this document, a receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and presents enhanced content.
SAP (Session Announcement Protocol): the protocol used for session announcements.
SDP (Session Description Protocol): SDP is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.
Transport operator: In the context of this document, the transport operator runs a video delivery infrastructure (terrestrial, cable, satellite, or other) that includes a transport for enhancement data.
Triggers: a text message used to announce the availability of an enhancement to the receiver, or execute ECMA script on a document in a running enhancement Triggers include the URL of the enhancement and optionally a human-readable string to use.in order to announce the availability of the interactive television experience to the user.
TV Enhancement: A collection of Web content displayed in conjunction with a TV broadcast as an enhanced or interactive program.
UDP (User Datagram Protocol): an Internet Standard transport layer protocol defined in STD 6, RFC 768. It is a connection-less protocol which adds a level of reliability and multiplexing to IP.
UHTTP (Unidirectional Hypertext Transfer Protocol): UHTTP is a simple, robust, one-way resource transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This resource transfer protocol is appropriate for IP multicast over television vertical blanking interval (IPVBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems.

UUID (Universally Unique Identifier) Also known as GUID (Globally Unique IDentifier) is an identifier that is unique across both space and time, with respect to the space of all UUIDs.

Proposed SMPTE Standard **SMPTE 364M**

Declarative Data Essence – Unidirectional Hypertext Transport Protocol

1 Scope

This document describes the Unidirectional Hypertext Transfer Protocol, or UHTTP. UHTTP is a simple, robust, one-way data transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This transfer protocol is appropriate for delivery of HTML and other content resources using IP multicast over television vertical blanking interval (IP/VBI) and other unidirectional transport systems.

2 References and Organization

2.1 Normative References

"The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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."

[UUID] ISO/IEC 11578:2000, Information technology, "Open Systems Interconnection – Remote Procedure Call", Annex A, "Universal Unique Identifier"

[RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, August 1998.

[RFC2068] Fielding, R., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

2.2 Informative References

[DDE1] SMPTE Proposed Standard 363M, "Declarative Data Essence, Content Level 1"

[DDEIPM] SMPTE Proposed Standard 357M, "Declarative Data Essence, IP Multicast Encapsulation".

[DDELID] SMPTE Draft Standard xxx, "The Local Identifier URL Scheme (lid:)"

[GZIP] Deutsch, P., "GZIP file format specification version 4.2", RFC 1952, May 1996.

[UMID] SMPTE Standard 330M-2000, "Television – Unique Material Identifier (UMID)"

[802] ANSI/IEEE 802-1990 "IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture"

2.3 Organization of this Document

3 Introduction

The Unidirectional Hypertext Transfer Protocol, or UHTTP, is a simple, robust data transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This transfer protocol is appropriate for delivery of HTML and other content resources using IP multicast over television vertical blanking interval (IP/VBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems. The UHTTP protocol is used in the Declarative Data Essence Content Level 1 specification [DDE1] to deliver television-related content resources (such as web pages, images and scripts) which were broadcast along with a television signal.

Resources sent using the UHTTP protocol are divided into a set of data segments encapsulated in UDP packets. Typically, these packets are delivered via multicast IP, but this is not required. When delivered via IP multicast, the address and port used must be exlusively for UHTTP. That is, other UDP-based protocol data would be indistinguishable from UHTTP data, and if combined, may result in unpredictable receiver behavior. Each packet contains enough header information to begin capturing the data at any time during the broadcast, even midway through

the transfer. This header contains a unique identifier (in the form of an UUID [Appendix A: UUID (Universally Unique Identifier)]) that uniquely identifies the transfer, and additional information that enables the receiver to place the data following the header in the appropriate location within the transfer. Additional header information indicates to the receiver how long to continue listening for additional data.

UHTTP includes the ability to gather segments over multiple retransmissions to correct for missing packets. It is also possible to group resources together for all-or-none delivery within a UHTTP transfer. The protocol also includes a simple forward error correcting mechanism which provides for the ability to restore missing data in the event of limited packet loss.

4 Data Transfer Features Enabled by the UHTTP Protocol

4.1 Robust Delivery: Gathering data over multiple transmissions

Data can be resent via UHTTP using the same globally unique TransferID. The data is delivered as individual segments, each of which is encapsulated within a UDP message, potentially delivered via IP multicast. Information in the header allows a receiving application to receive segments out of order or multiple times. If the transfer data is sent repeatedly, the receiving service can fill in missing ranges using these retransmissions. This provides robust (though not necessarily reliable) data delivery. Additionally, forward error correction (FEC), using an exclusive-or-based algorithm, provides for recovery of some missing segments in the face of segment loss without retransmission.

4.2 Meta-information in the form of HTTP-style headers

The protocol provides for the inclusion of HTTP-style headers preceding the resource data. These headers may include information describing the content type of the resource and content location in the form of a URL. It may also be used to describe groups of resources as a multipart construction. Other meta-information, including date stamping and expiration dates, may be usedto provide additional information about the resource content.

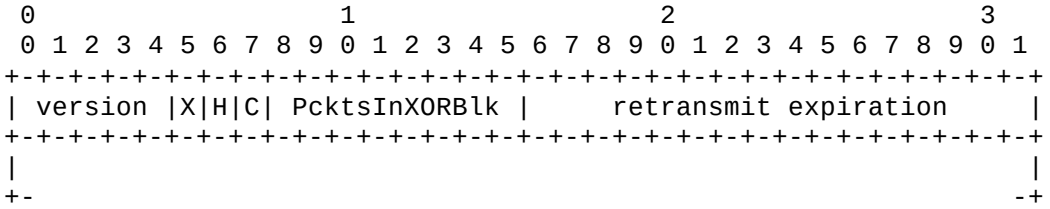
5 UHTTP Header Format

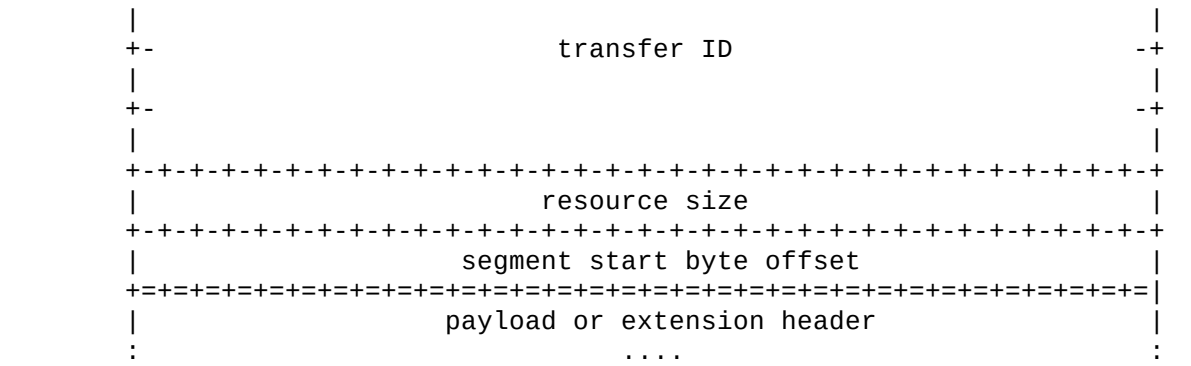
5.1 Basic UHTTP Header Format

This section describes the format of the message packets that carry UHTTP data. It describes the information needed to create the messages using the protocol on the broadcast side and to turn those messages back into resources on the receiving side.

The UHTTP header is at the start of every UHTTP IP/UDP datagram. All values are network byte order.

The UHTTP header has the following format:





The fields are described below.

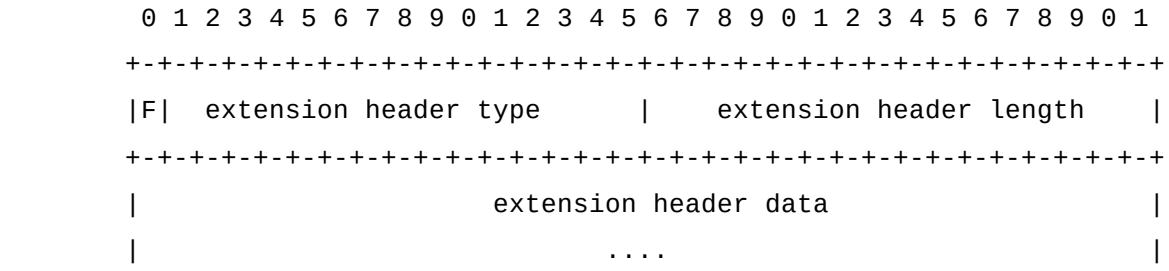
Name	Size	Description
Version	5 bits	Describes the version of the protocol. The protocol described here is version 0.
ExtensionHeader (X)	1 bit	When set, this bit indicates that one or more extension header fields are present and immediately follow the main UHTTP header.
HTTPHeadersPrecede (H)	1 bit	A bit flag that, when set to 1, indicates that HTTP-style headers precede the resource data. These HTTP-style headers are considered part of the data when calculating the ResourceSize and SegStartByte fields, as well as for forward error correction. This bit must be set in all packets associated with a UHTTP transfer when HTTP-style headers precede the data. When set to zero, no HTTP-style headers precede the resource data.
CRCFollows (C)	1 bit	When the CRCFollows bit is set to 1, a 32-bit CRC is calculated and can be used to detect possible corruption in the data delivered via UHTTP. Using the MPEG-2 CRC algorithm, the CRC is calculated on the complete data, including HTTP-style headers, if any. It is then appended to the end of the data in the last logical packet. This CRC field is considered part of the data for the purposes of calculating the resource length and calculating the forward error correction. The bit must be set in all packets associated with a UHTTP transfer when a CRC is used.
PacketsInXORBlock (PcktsInXORBlk)	1 byte	Describes the number of packets in a forward error correction block, including the forward error correction packet. Set to zero when no forward error correction is used. See Section 4 for details on the forward error correction mechanism.
RetransmitExpiration	2 bytes	Describes the time in seconds over which the resource may be retransmitted. This indicates how long the

		receiving software should wait to try to recover missing packets that follow in retransmissions of the same resource. This allows a resource to be carouseled, or sent repeatedly to increase the chances of delivery without missing segments. Set to zero if the resource will not be retransmitted. Set to maximum if the software should continue listening. The RetransmissionExpiration field should be updated to remain accurate during retransmissions, including the current transmission.
`TransferID`	`16 bytes`	Globally unique identifier (UUID [10]) for the UHTTP transfer. This ID allows receiving software to identify which segments correspond to a given transfer, and determine when retransmission occurs.
`ResourceSize`	`4 bytes`	Size of the complete resource data itself (excluding segment headers, XOR segments and padding for exclusive-or correction). This length does include the length of the HTTP-style headers, if any, as well as the 4-byte CRC, if the CRCFollows bit is set to 1.
`SegmentStartByteOffset (SegStartByte)`	`4 bytes`	Start offset for the first byte in the transfer for this data segment. When XOR data is used to replace missing packets, SegStartByte includes the XOR data as well as the resource data, and optional HTTP-style headers and CRC. This allows for determining where all packets fit regardless of delivery order. The exclusive-or correction packet looks like any other UHTTP packet. Its data payload is simply the exclusive-or of a number of packets that precede it in order in the data. The number of packets in an XOR block is specified in the PacketsInXORBlock field described above. The UDP packet data length for the enclosing UDP packet is used to determine the length of the segment. It is permissible to send a packet that contains UHTTP header (and optional extension headers), but without any data. If no data is included, then the SegStartByte field is ignored.

5.2 UHTTP Extension Header Format

If the ExtensionHeader flag is set in a UHTTP packet header, additional optional header fields are present. These fields appear directly after main UHTTP header. Extension headers are optional on a packet-by-packet basis, and may appear on none, some or all of the UHTTP packets transmitted, depending on the ExtensionHeaderType. This specification defines a single extension header type, HTTPHeaderMap (defined below). It is assumed that receivers ignore any extension headers with an unknown type.

When the Extension Header (X) is set to a value of one, one or more extension headers may follow the UHTTP header and precede the data segment in that packet. Extension headers have the following format:



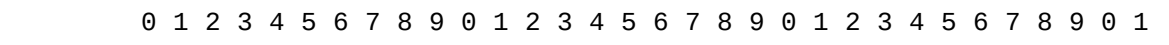
Name	Size	Description
ExtensionHeaderFollows (F)	1 bit	When set to 1, this field indicates that another extension header follows this one. When set to 0, the UHTTP data payload follows this extension header.
ExtensionHeaderType	15 bits	Identifies the extension header type. (1 = HTTPHeaderMap, all other values reserved).
ExtensionHeaderDataSize	2 bytes	Describes the length of the complete Extension Header data in bytes. Zero indicates that there is no ExtensionHeaderData following.
ExtensionHeaderData		The variable length data for this extension header. The length of the ExtensionHeaderData field is indicated by the ExtensionHeaderDataSize.

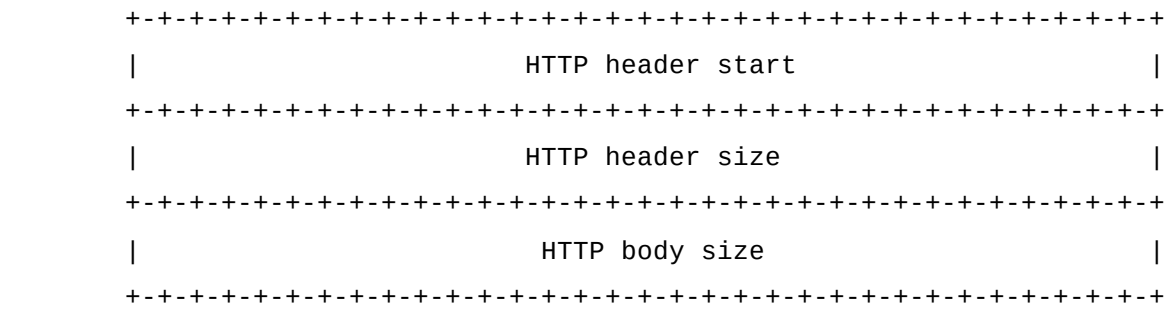
If the ExtensionHeaderFollows bit is set, then another ExtensionHeader follows this header. If the bit is cleared, then the UHTTP data payload follows the ExtensionHeaderData (if any) immediately.

5.2.1 HTTPHeaderMap Extension Header

One ExtensionHeaderType is defined for this specification. When ExtensionHeaderType is set to a value of 1, then the ExtensionHeaderData field contains an HTTPHeaderMap. A HTTPHeaderMap extension header may optionally be included whenever the UHTTP transfer contains HTTP-style header information (as indicated by the HTTPHeadersPrecede bit in the main UHTTP header). If HTTPHeaderMap extension headers are used, they should be included in every packet in a UHTTP transfer that contains header, body or forward error correction (FEC) data.

The HTTPHeaderMap consists one or more sets of the following fields:





Name	Size	Description
HTTPHeaderStart	4 bytes	This field indicates an offset into the UHTTP data, in bytes, where a HTTP-style header is found. The offset is calculated from the beginning of the corrected UHTTP data, and does not include the FEC data when the FEC mechanism is used.
HTTPHeaderSize	4 bytes	This field indicates the length of the HTTP-style header, in bytes, including the HTTP-style header fields, the terminating pair of carriage-return/newline character sequences, and any preceding multipart boundary lines.
HTTPBodySize	4 bytes	This field indicates the length of the data body, in bytes, associated with the HTTP header described in this map entry.

When the UHTTP transfer consists of a single (i.e. non-multipart) resource, a single 12 bytes set of HTTPHeaderMap fields is present in the HTTPHeaderMap. The HTTPHeaderStart, in this case, will be set to zero and the HTTPHeaderSize will be set to the sum of the length of the HTTP-style header fields and all separating carriage-return/newline characters. The HTTPBodySize field will contain the size, in bytes, of the body data related to that header field.

When a UHTTP transfer contains multiple resources (as specified by a multipart content-type), multiple sets of HTTPHeaderMap groups may be included in the HTTPHeaderMap data, each indicating the offset and size of the HTTP-style headers for each resource, (including any multipart boundary lines, HTTP-style header fields and separating carriage-return/newline characters), as well as the size of the body relating to each header.

When including HTTPHeaderMap data, senders must at a minimum include HTTPHeaderMap entries for each HTTP-style header that is partially or completely included in a given packet. Additionally, when forward error correction is used in UHTTP transfers that contain HTTPHeaderMaps extension headers, senders must include HTTPHeaderMap entries as extension headers in FEC-data packets for all HTTP-style header sections that may be corrected by the FEC packet. Senders are free to include additional HTTPHeaderMap entries in any packet beyond the minimum.

6 Forward Error Correction Mechanism

In addition to the ability to retransmit data and have missing segments filled in during retransmissions, this transfer protocol can also include extra data packets that can be used for simple missing packet error correction. When the PacketsInXORBlock header field is set to zero, there is no exclusive-or forward error correction. When non-zero, forward error correction packets may be sent to allow for correction of missing or corrupted packets by the receiver, and all segments must be of the same length. It is permissible to send packets with no data payload (but with UHTTP headers and optional extension headers). In this case, the packet is ignored in the calculation of forward error correction.

In blocks of PacketsInXORBlock equal size data segments, the last data segment in the block contains the exclusive-or of the preceding segments (PacketsInXORBlock - 1). Each byte of the data in this "XOR segment" is the exclusive-or of the corresponding byte in each of the other segments in that data block. If the data is thought of as laid out separated into consecutive segments, then after every PacketsInXORBlock - 1 segments another segment is inserted that looks exactly like resource data and has its own position offset into the transfer like resource data. The data in that segment is the exclusive-or of the previous packets in that block. XOR data in the XOR packet is the exclusive-or of data segment contents only, including the HTTP-style header fields but not including the UHTTP header that is also in the packet.

If forward error correction is used, the data payload of all packets must be the same size. The packet containing the data at the end of the file (including the optional CRC) must be zero filled. Packets between this packet and the last XOR packet need not be sent since it is assumed that the receiver knows their contents are all zeros, as it was provided the overall length in the UHTTP resource size field. If they are sent they must be zero filled after the segment header; if not sent, they are assumed to contain a payload of all zero bytes for the purposes of forward error correction calculations. The last XOR packet has the value SegStartByte calculated to be just as if zero-filled extra packets were sent, but there is no requirement to send those empty packets.

To correct for a single missing packet in a block, it is assumed that the receiver calculates the exclusive-or the data payload of the packets that arrived with the XOR data segment for that block. An important point is that segments can be sent in any order since each segment including the XOR segment indicates where in order they belong. By sending segments (including the XOR packets) out of order, there may be protection against burst errors that lose successive packets. When re-transmitting a UHTTP transfer, a different order of segments can be used in each retransmission to avoid different types of burst errors. This protocol allows the headend (broadcast side) tools to decide how to order sending packets, thereby providing a great deal of flexibility. The receiving side does not need to know the transmission order (consistent with the fact that in general it cannot know the transmission order for IP multicast delivery).

7 HTTP-style headers used in UHTTP

The UHTTP transfer protocol can be used to deliver resources via a broadcast medium, which can simultaneously deliver resources, including web-related content, to large numbers of users simultaneously. HTTP-style header fields can be included in the UHTTP data to provide meta-information about the content, including identifying URIs, expiration dates and content type and encoding. The use of HTTP-style headers is optional in UHTTP, but is required for resources intended to be interpreted as web content.

Note: While HTTP headers preceding resource content are optional in the UHTTP protocol, they are required when the protocol is used for delivery of Declarative Data Essence [DDE1] content.

HTTP-style headers, as specified by HTTP 1.1[RFC2068], precede the resource content just as in HTTP transfers when resources are sent as a response to a HTTP GET or POST command. The HTTP-style headers may provide additional information to the browser, for example, the expiration time for the resource. The HTTP-style headers precede the body of the resource data, and are treated as part of the data payload. The protocol header and its version imply the equivalent HTTP response line (e.g. "HTTP/1.1 200 OK"). The header fields that are expected to be supported by all receivers are listed below and should be interpreted per the HTTP 1.1[RFC2068] specification. No assumption should be made for support of other header fields.

7.1 Supported HTTP-style headers

Header fields required for every resource when using HTTP-style headers:

```
Content-Length:
Content-Location:
```


Recommended HTTP-style header fields:

```
Content-Type:
```

Optional HTTP-style header fields:

```
Content-Base:
Content-Encoding:
Content-Language:
Content-Style-Type:
Date:
Expires:
Last-Modified:
```

UHTTP transfers that utilize HTTP-style headers MUST contain the required headers listed above ("Content-location" and "Content-length"), to ensure that an appropriate URI is specified for the resource and to ensure that the resource data is delivered in whole. No other header fields are strictly required, but may provide useful information to the receiver in determining the disposition of the content.

Note: It is assumed that receivers compliant with the Declarative Data Essence Content [DDE1] and IP Multicast Encapsulation [DDEIPM] specifications support “gzip” [GZIP] content encoding, as specified by the “`Content-Encoding`” HTTP header field.

It is assumed that receivers will decode the headers and data and store them in a local cache system, and that different receiver platforms will have different cache sizes for storing local resources, which may or may not correspond to traditional web browser caches. Resources transmitted with "Content-location" header fields, which contain “http:” URLs, identify resources, which are also available via an HTTP request specified by that URL. The use of "Content-Location:" headers with local identifier [DDELID], or "lid:" URLs is intended to mirror resource delivery to a local cache without requiring that the data be available on the Internet.

It is assumed that receiving platforms should take into consideration that the same resources will likely be sent repeatedly to provide resources for users who tune in late. HTTP-style header fields can be examined by receivers to determine if the resource is already present, and so can be ignored. The "Date:", "Expires:", and "Last-Modified:" headers can be used by receivers to determine the lifetime of a resource in a given receiver's cache.

8 Packaging more than one resource

It is possible to send multiple resources in a single UHTTP transfer by packaging them into a single multipart resource for all-or-nothing transfer. In this case, the HTTP-style "Content-Type:" header field would have a value of "multipart/related"[RFC2387]. When this type is used, the HTTP-style header is ended as usual and is followed by the usual boundary structure for "multipart/related" separating multiple related resources that each use the HTTP-style header formats. This is a mechanism to package multiple related resources together in a single all-or-nothing transfer. The HTTP-style headers for individual sub-parts describe only that sub-part, and are interpreted as per the HTTP 1.1 specification [RFC2068]. In this case, it may be convenient to specify a "Content-Base:" for the entire package and then specify relative URLs for each of the "Content-Location:" headers for subsequent subparts. The "multipart/related" content type should be used as per RFC 2387[RFC2387].

An example using HTTP scheme URLs: (Informative)

```
Content-Base: http://www.blahblah.com/
Content-Length: 3495
Content-Type: Multipart/Related; boundary=example98203804805
--example98203804805
Content-Location: resource1.html
Content-Length: 495
Content-Type: text/html

...Resource data for resource1.html...
...<IMG src="image.jpg">...
```

```
--example98203804805
Content-Location: image1.jpg
Content-Length: 1495
Content-Type: image/jpeg

...Resource data for image1.jpg...
--example98203804805
```

9 Security Considerations

There are a number of security issues associated with the use of UHTTP to deliver content on public broadcast networks, many of which are shared with electronic mail systems. When HTTP-style headers are used to identify the resources in a UHTTP transfer, it is possible for the sender to misrepresent the content with URIs, which do not match the transmitted content. The security issues associated with this mislabeling, as well as the security issues associated with the use of HTML content which is broadcast are the same as those identified in section 11.1 of RFC 2387[RFC2387]. Additionally, there are security issues associated with the caching of data transmitted via UHTTP which are the same as those identified in section 11.2 of RFC 2387[RFC2387].

It should be noted that many broadcast systems employ conditional access systems (satellite and cable television, for example), which provides a level of security for the broadcast channel leading up to the receiver. Unfortunately, it may be possible to insert UHTTP data earlier in the broadcast chain at points which are less secure (on videotape to be played into the system, for example), so applications using UHTTP may which cannot ensure end-to-end security of the broadcast system should employ additional security mechanisms.

10 Appendix A: UUID (Universally Unique Identifier)

Definintion: UUID (Universally Unique Identifier) Also known as GUID (Globally Unique IDentifier) is a fixed-size 128-bit identifier that is unique across both space and time, with respect to the space of all UUIDs.

The generation of UUIDs does not require that a registration authority be contacted for each identifier. Instead, it requires a unique value over space for each UUID generator. This spatially unique value is specified as an IEEE 802 [802] address, which is usually already available to network-connected systems. This 48-bit address can be assigned based on an address block obtained through the IEEE registration authority.

When IEEE 802 address is available to a system desiring to generate a UUID, the method described in ISO 11578, Annex A [UUID] shall be used. If no IEEE 802 address is available, other methods that provide a way to generate a probabilistically unique one that cannot conflict with any properly assigned IEEE 802 address, shall be used. The SMPTE Standard 330M-200

[UMID], for example, describes a method similar to the ISO method, but more suited to television systems where some of the ISO fields were not easily created.

Proposed SMPTE Standard SMPTE 361M	
NTSC IP and Trigger Binding to VBI	

1 Scope

This document defines a standard manner for the carriage of Declarative Data Essence [DDE1] triggers and IP packets in the vertical blanking interval of the NTSC video standard.

2 References and Organization

2.1 Normative References

[NTSC] SMPTE 170M, "NTSC"

[RFC2728] IETF RFC 2728, "The Transmission of IP Over the Vertical Blanking Interval of a Television Signal"

[EIA746] EIA-746A, "Transport Of Internet Uniform Resource Locator (URL) Information Using TEXT-2 (T-2) Service"

[EIA608] EIA-608, "Recommended Practice For Line 21 Data Service "

[EIA516] EIA-516, "NABTS"

[DDE1] SMPTE Proposed Standard 363M, "Declarative Data Essence, Content Level 1"

[DDEIPM] SMPTE Proposed Standard 357M, "Declarative Data Essence, IP Multicast Encapsulation"

3 Introduction

In NTSC, IP data is broadcast by encoding bytes in the vertical blanking interval of individual video fields. Two different techniques are used for broadcasting data using transport A and transport B as defined in the content level specification [DDE1].

4 VBI Line 21

Triggers are transmitted on VBI Line 21 of the NTSC signal using the T-2 service as specified in EIA-608. This encoding is consistent with the EIA-746A specification which describes how to send URLs and related information on VBI line 21 of an NTSC channel, without interfering with other data (e.g., closed captions) also sent on that line. The checksum described in the DDE trigger definition [DDE1] is required in the Transport A binding to NTSC.

Note that, as specified in the trigger definition, triggers are encoded using ISO-8859-1 [DDE1] and not the EIA-608 character set. (While most characters are the same in both encodings, a few codes have different meanings.)

Trigger length should be kept as short as possible. Trigger transmissions should be limited to 25% of the total field 1 bandwidth, even if more bandwidth is available after captioning, to allow for other downstream services.

5 IP over VBI

IP datagrams shall be sent according to IETF RFC 2728. In NTSC, the NABTS [NABTS] (rather than WST) byte encoding shall be used.

IP streams shall be sent on the NABTS packet addresses 0x4b0 through 0x4bf. Other packet addresses may be used, but receivers are only required to handle IP datagrams arriving using packet addresses 0x4b0 through 0x4bf.

Proposed SMPTE Recommended Practice SMPTE XXXX PAL/SECAM IP AND TRIGGER BINDING TO VBI	D27.110-2397B

1 Scope

This document defines a standard manner for the delivery of SMPTE DDE content for both Transport A and Transport B for 625 lines television system. Both transport types are based on carriage of IP multicast packets in VBI lines of a PAL/SECAM system.

2 References and Organization

2.1 Normative References

The following standards contain provisions that through reference in this text constitute provision of this standard. At the time of publications, the editions 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.

[RFC1055] IETF RFC 1055, "A nonstandard for transmission of IP datagrams over serial lines: slip".

[ETS300706] ETS-300706, "Enhanced Teletext specification".

[ETS300708] ETS-300708, "Data transmission within Teletext" (edition 2).

[TR-054] EACEM TR-054, "Register of Application Codes for Teletext based systems", EACEM Technical Report, January 2001. *(ETS number not yet assigned)*

[DDE1] SMPTE Standard 343M, "Declarative Data Essence, Content Level 1".

[IPM] SMPTE Standard 357M, "Declarative Data Essence, IP Multicast Encapsulation".

[IANA] (reference to registered values when issued)

3 SMPTE DDE binding on PAL/SECAM system (625 line television system)

In PAL/SECAM systems (625 line television systems), SMPTE DDE content [DDE1] shall be retrieved either using Transport A (broadcast of triggers and pulling of data by the return channel) or using Transport B (broadcast of triggers and data).

Transport B is described in [IPM] using IP encapsulation. For transport A, each single trigger is carried in a single UDP/IP multicast packet and these IP multicast packets are delivered on the fixed IP multicast address [TBD] and port [TBD] as defined in [IANA].

For both transport type, IP multicast data is broadcast by encoding bytes in the vertical blanking interval of individual video fields using the protocol stack shown in figure 1. To allow a receiver

to recover IP packets, successive packets are delimited using SLIP [RFC1055] as a framing protocol. Data are transported in teletext packet with an address of 30 or 31 as defined by the IDL Format B specification [ETS300708].

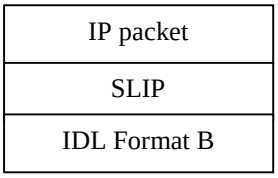


Figure 1: Protocol stack used to transmit IP multicast packets.

The Independent Data Line format B is defined in clause 6 of the second edition of [ETS300708]. IDLs are teletext packets with an address of 30 or 31 (see [ETS300706] for the definition of a teletext packet). The general structure of IDL-B is presented in Figure 2.

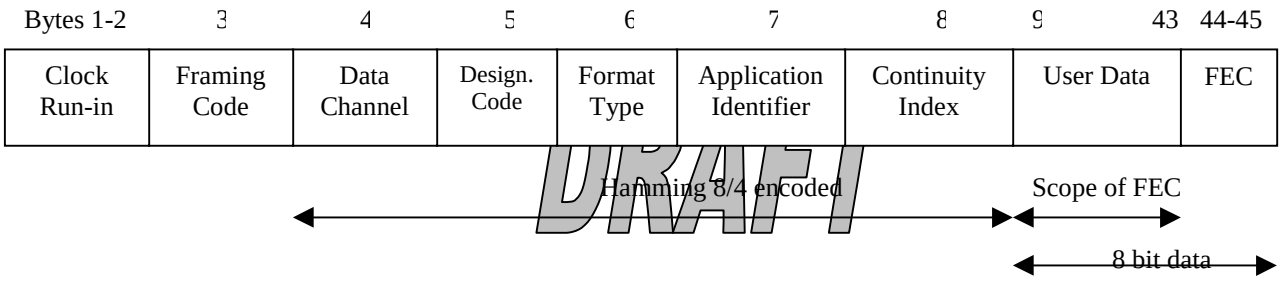


Figure 2: Structure of an IDL Format B packet.

For the purpose of this specification, the different component fields of an IDL-B are defined as follows:

- Clock Run-in: As defined in [ETS300706].
- Framing Code: As defined in [ETS300706].
- Data channel: As defined in subclause 6.4 of [ETS300708].
- Designation Code: As defined in subclause 6.4 of [ETS300708].
- Format Type: The four message bits of the Hamming 8/4 encoded Format Type byte is 0001.
- Application Identifier: The four message bits of this filed is Hamming 8/4 encoded. The list of possible value is recorded in TR-054. The value of this four message bits is either application dependant (for example the value 0101 is reserved for TAK) or fixed to 1000 [TR-054]. The latter four message bits is intended to be used when IP datagrams transporting Declarative Data Essence [DDE1] content are supposed to be received by different kind of platforms.
- Continuity index: As defined in [ETS300708].
- User Data: This field carries a data stream made of IP datagrams delimited using the SLIP protocol. This protocol describes in [RFC1055] makes special use of the code values 0xC0 and 0xDB: The code 0xC0 is used as a delimiter between IP packets and shall not appear in the data stream anywhere else. It shall be inserted immediately before

the start of any IP packet. If a data byte within a IP packet has the code value 0xC0, it shall be replaced by the two-byte sequence 0xDB, 0xDC. If a data byte within IP packet has the code value 0xDB, it shall be replaced by the two-byte sequence 0xDB, 0xDD. Any number of 0xC0 bytes may be inserted between IP packets. As a minimum, a single instance shall be inserted between successive IP packets.

- FEC: To ensure the integrity of the data, a Forward Error Correcting scheme based on the standard t=1 Reed Solomon encoding is used as described in [ETS300708]