INTERNET DRAFT

Expires in six months

Paul Burchard,
Princeton CS
Dave Raggett, W3
Consortium

Compound Documents in HTML

<draft-ietf-html-cda-00.txt>

Status of this Memo

1. Abstract

This specification provides an HTML implementation of a simple compound document architecture for the World Wide Web, based on a new <EMBED> element.

By not restricting itself to a limited class of media types or media handler implementations, this element enables portable compound document markup, and encourages the modular design of user-agents. Although this specification does not presume to define a concrete API between extensible user-agents and their media handlers, some high-level requirements are imposed on the embedding semantics in order to ensure support for the full linking and embedding model.

By making <EMBED> a container element, rich alternative text with links and images is enabled. Moreover, the container element provides superior extensibility, setting the stage for structured enhancement of SGML content models, rather than sole dependence on proliferation of attributes.

Compound Documents in HTML

Contents

2. Examples and Rationale

The only compound document feature defined in HTML 2.0 [1] was the <IMG> tag for embedding image media into HTML. Although arguably, next to <A>, the single most influential tag in the explosive growth of the Web, its shortcomings are now causing a proliferation of media- and user-agent-specific attempts at extending HTML.

The most serious shortcoming of the <IMG> tag is its arbitrary restriction to image media. Many of the proposed embedding extensions also use names which suggest specialized functionality (e.g. <APPLET>, <FIG>). But as modular, extensible user agents become the norm, such restrictions become untenable. We propose that the name EMBED is both broad and intuitive enough to denote a generic embedding element. In order to avoid implementation-dependent markup, it is essential that this EMBED tag should cover all embeddable Internet media types [2], however the

corresponding media handlers are implemented (`built-ins', `plug-ins', etc.).

The second key shortcoming of the <IMG> tag is that it is `empty' (meaning that it doesn't use a closing </IMG>). Since SGML attributes are the only mechanism then available for the expression of element properties, all the powerful structuring capabilities of SGML are lost. In particular, the inability to nest elements inside an empty element means that properties with rich text values (such as would be desirable for the alternative text of <IMG>) cannot be implemented.

Moreover, extensibility of empty elements suffers because everything occurs in the a single, flat attribute namespace. Whereas empty

elements with complex properties create an "attribute soup" that may not even be legal SGML:

```
        <IMG APPLETSRC="my.applet" ALT="[applet]" FOO="23.8"
BAR="5.0" ...>
        <!-- WRONG: cannot pass arbitrary attributes -->
```

complex container elements can grow in a natural way, benefitting from the structured design of SGML:

```
        <EMBED SRC="my.applet">
                <PARAM NAME="foo" VALUE="23.8">
                <PARAM NAME="bar" VALUE="5.0">
                ...
                <em>Download</em> this cool <A
HREF="my.exe">application</A>!
                <!-- rich alternative text -->
        </EMBED>
```

The ability to pass an extensible set of parameters to the embedded media handler in this way is one of the main requirements for an embedding tag.

Notice that this structured content approach also makes it possible to combine <EMBED> with <IMG> for excellent backwards compatibility:

```
        <EMBED SRC="movie.mpg">
                <IMG SRC="movie.001.jpg"> <!-- first frame -->
        </EMBED>
```

Here, an image of the first frame of the movie will be shown whenever the movie itself is not displayed, providing fallback for all of the following situations:

* the user-agent does not know about the <EMBED> tag, or

* the user-agent does not have an embeddable handler for the video/mpeg media type, or

*   the linking and embedding fails in any other way.

3. Compound Document Architecture

Because the Web, as a distributed hypermedia system, makes a clear separation between storage and presentation of data, the model described in this section would more accurately be called a "compound presentation architecture". The model offered here takes a very high-level view of client-side state, and explains how links, the fundamental building blocks of the Web, are used to assemble compound presentations in stateful user-agents.

The model views the user-agent as a repository of hierarchical presentation resources. A presentation resource is an addressable unit of state in the user-agent, such as the document display area of a GUI window. An embedded display area (say, showing a video) within the larger display area (say, showing an HTML document) would be a child in the presentation resource hierarchy. This hierarchical structure makes it possible to use URLs (within the generic-RL paradigm [3]) as addresses, but this document does not define or require any specific client-side addressing schemes.

Like server-side resources on the Web, these presentation resources may support methods to query and change the state they hold. This model views the standard display of a MIME entity in a presentation resource as POSTing that entity to the resource. The implementation of the POST method typically creates a media handler for the entity (which does the actual work of display), pushes that media handler onto a navigation stack, and then gives the media handler access to the display area represented by the resource.

This model of the user-agent now allows us to describe an anchor more generally as a recipe for getting a MIME entity from a retrieval resource and posting it to a presentation resource. A link, as always, is a relation between two anchors (known as the tail and the head). Thus, links and anchors have the following properties:

```
link:
    head anchor, tail anchor, relations

anchor:
    source method call:
        retrieval address, method, and argument(s)

    target method call:
        presentation address, method, and argument(s)
```

For this specification, the main point of the model is to prescribe what information must be made available to the media handlers by the

presentation resource (this affects the design of `plug-in' APIs). The key requirement we impose is that the media handler must have acccess to complete link information according to the link model just described.

4. Geometry Negotiation

The actual appearance of the compound presentation depends on a negotiation between the containing and contained presentations. This negotiation process may involve:

* the respective style sheets of the entities presented

* the nature of the media involved (resizable or not)

[This section is incomplete.]

5. HTML Markup

This specification defines an extension of the HTML 2.1 DTD [4]. This description makes free use of SGML elements and entities defined there.

5.1. Elements

```
<!ELEMENT EMBED - - (PARAM*, CAPTION?, EMBED.BODY, CREDIT?)>
<!ATTLIST EMBED
     %attrs;
     %link.internal;
     %link.metainfo;
     %case.metainfo;
     %size;                    -- size of reserved area --
     %align;                   -- alignment or float --
     %SDAPREF; '<Fig><Xref IDRef="#AttVal(SRC)"><?SDATrans
Embed: #AttList>'
     %SDASUFF; '</Fig>'
     >
```

Attributes: SRC, PARAMS; TITLE, URN, REL, REV; ACCEPT, ACCEPT-CHARSET, ACCEPT-ENCODING; WIDTH, HEIGHT; ALIGN, HSPACE, VSPACE, FLOWTO.

Common Attributes: ID, LANG, DIR (implied for all elements below).

From the point of view of the compound document architecture, the <EMBED> element has two purposes:

* It conditionally creates a presentation resource, a child node in the presentation hierarchy below the presentation of the HTML entity which contained the <EMBED> element.

* It declares a link for which the target of the tail anchor implicitly uses the newly created presentation resource.

This functionality is implemented with the help of the <EMBED.BODY> and <PARAM> elements:

```
<!ELEMENT EMBED.BODY O O %A.content>
<!ATTLIST EMBED.BODY
     %attrs;
     >
```

The <EMBED> link is activated as soon as the parent presentation of the HTML entity is created. If for any reason the link fails, or the child presentation cannot be created, then the content of the <EMBED.BODY> element must be rendered in place of the <EMBED> element.

Note: There is never any need to actually include <EMBED.BODY> tags in a document; this construct exists solely to allow glitch-free use of %A.content in combination with <PARAM> tags. The %A.content was chosen in part to aid error recovery when the </EMBED> tag is accidentally omitted.

```
<!ELEMENT PARAM - O EMPTY  -- builds presentation specializer -- >
<!ATTLIST PARAM
     %attrs;
```

```
        %key.value;
        %SDAPREF; "<?SDATrans Param: #AttList>"
        >
```

Attributes: NAME, VALUE, ACCEPT, ACCEPT-CHARSET, ACCEPT-ENCODING.

The method invoked on the presentation resource is POST, which has two arguments. The first is the MIME [5] entity retrieved as a result of link activation. The second argument is a MIME entity called the presentation specializer, which is used to modify the resulting presentation, and is a generalization of the so-called "fragment id". This specializer can be defined in one of three ways, in order of priority:

<PARAM> elements
: these elements collectively define a presentation specializer of type multipart/form-data, which is used by the media handler as

: an unordered list of named parameters, each with full MIME type information

PARAMS attribute
: defines a presentation specializer of type application/x-www-form-urlencoded, which is used by the media handler as an unordered list of named string parameters

"fragment id" of SRC attribute
: defines a presentation specializer of type text/plain, which used as the name of the markup element on which the presentation should be focused

If two or more of the above mechanisms is used simultaneously, the one with higher priority wins.

```
<!ELEMENT INPUT - O EMPTY  -- builds retrieval specializer -- >
<!ATTLIST INPUT
     %attrs;
     %key.value;                    -- key and MIME-typed value --
     TYPE       %InputType  #IMPLIED  -- defaults to TEXT in
FORM context --
     CHECKED    (CHECKED)   #IMPLIED  -- initial boolean state --
     SRC        %URI;       #IMPLIED  -- embedded graphic for
```

```
TYPE=IMAGE --
        %align.simple;                   -- alignment for TYPE=IMAGE --
        SIZE       CDATA       #IMPLIED
        MAXLENGTH  NUMBER      #IMPLIED
        %SDAPREF; "Input #AttVal(Type): "
        >
```

Attributes: NAME, VALUE, ACCEPT, ACCEPT-CHARSET, ACCEPT-ENCODING; TYPE; CHECKED; SRC; ALIGN, SIZE, MAXLENGTH.

The <INPUT> element plays a largely parallel role to <PARAM>, but on the retrieval (source) end of the anchor rather than the presentation (target) end. Unfortunately, this element has been somewhat abused as a result of its great usefulness (though fixing these problems is beyond the scope of this proposal).

The only modification specified here is the addition of %mime.constraints; type information for the value of the input field (described in detail below). Note that the %mime.constraints; information is not intended to apply to the vestigial embedding link defined by the SRC attribute.

```
   <!ELEMENT CAPTION - - (%text;)+ -- caption for floating element -->
   <!ATTLIST CAPTION
        %attrs;
        %align;                     -- side of rectangle where
caption placed --
```



```
        %SDAPREF; "Caption: "
        >
```

Attributes: ALIGN, HSPACE, VSPACE, FLOWTO.

```
   <!ELEMENT CREDIT - - (%text;)* -- copyright/credit for embedded
object  -->
   <!ATTLIST CREDIT
```

```
        %attrs;
        %SDAFORM; "Fn"
        >
```

In typical rendering with default attributes, the italicized, centered caption would be placed at the bottom of the area reserved for the child presentation, and the credit would appear as smaller roman text right-aligned near the bottom of the reserved area.

Because the caption is not an independent float, the ALIGN attribute must be interpreted somewhat differently. Here, the value of ALIGN indicates on which side of the reserved area the caption should be placed. The values MIDDLE and CENTER are not meaningful.

```
   <![ %HTML.Highlighting [
   <!ENTITY % text
"#PCDATA|A|EMBED|IMG|BR|%phrase|%font|SPAN|Q|BDO|SUP|SUB">
   ]]>
   <!ENTITY % text "#PCDATA|A|EMBED|IMG|BR|SPAN|Q|BDO|SUP|SUB">
```

The <EMBED> tag is permitted as part of %text context, meaning that it does not force a paragraph break. The reason is that <EMBED> functions as either a float (alongside the text flow) or a glyph (part of the line of text line), but never as a block.

5.2. Attribute Value Types

HTML is an application of SGML for which application conventions play a strong role, particularly in the rich syntax of its attribute values. Although these conventions cannot be checked directly by the SGML parser, DTD "macros" can be used help flag these conventions for other validation tools.

```
   <!ENTITY % Length    "CDATA"    -- number followed by optional
units -->
```

Attributes values defining a length are uniformly specified as a number followed by an optional suffix denoting the units of measurement. The number must be an integer value or a floating-point real value such as 2.5 (scientific notation, as in 1.2e2, is not allowed). No white space is allowed between the number and the suffix. The default units are screen pixels, and the following suffixes may also be recognized: pt denotes points, pi denotes picas, in denotes inches, cm denotes centimeters, mm denotes millimeters, em denotes em units (in the default font), and px denotes screen pixels.

```
<!ENTITY % URI       "CDATA"    -- uniform resource identifier -->
```

The syntax of Uniform Resource Identifiers is given by RFC 1630 [6]. For historical reasons, attribute values of type %URI which identify retrieval resources may also include a final "fragment identifier", which is actually unrelated to the retrieval process and instead specifies a presentation specializer of type text/plain.

5.3. Attribute Sets

The HTML DTD does not make extensive use elements for structuring; instead it is typified by sets of related attributes which recur in multiple elements. In such a setting, DTD evolution can be facilitated by making consistent use of SGML "macros" to identify and reuse such attribute sets.

```
<!ENTITY % attrs
     'ID      ID         #IMPLIED   -- element identifier --
      LANG    NAME       #IMPLIED   -- RFC 1766 language tag --
      DIR     (ltr|rtl)  #IMPLIED   -- text directionality --'>
```

The %attrs; attribute set is common to all elements.

ID
: Used to define a document-wide identifier. This can be used for naming positions within documents as the destination of a hypertext link. An ID attribute value is an SGML NAME token. (NAME tokens are formed by an initial letter followed by letters, digits, "-" and "." characters. The letters are restricted to A-Z and a-z.)

LANG

DIR

These attributes are described in the HTML 2.1 specification

[4].

```
<!ENTITY % link.internal  -- links with implicit presentation
resource --
        'SRC     %URI;      #REQUIRED  -- resource to retrieve --
         PARAMS  CDATA      #IMPLIED   -- presentation specializer --'>
```

SRC
The address of the resource to be retrieved in traversing this link. The address must be a valid URI [6].

PARAMS
Declares a presentation method specializer argument of type application/x-www-form-urlencoded [1]. (It is strongly recommend to use `;' as a separator in place of `&'.) This attribute is suitable for declaring small, simple parameter lists; more general presentation specializers can be created using the <PARAM> mechanism.

```
<!ENTITY % mime.constraints  -- MIME typing constraints --
      'ACCEPT           CDATA  "text/plain" -- applicable media
type(s) --
         ACCEPT-CHARSET   CDATA  "ISO-8859-1" -- appl. character
encoding(s) --
         ACCEPT-ENCODING  CDATA  "ISO-8859-1" -- appl. transfer
encoding(s) --'>
```

ACCEPT
ACCEPT-CHARSET
ACCEPT-ENCODING
In link-related elements, the %mime.constraints attribute-set provides an advisory range of MIME type information, suggesting to the data producer what is acceptable to the consumer, and to the consumer what is available from the producer. This attribute-set also provides specific defaults for situations in which the MIME type information of the corresponding data cannot otherwise be determined.

The attribute values have syntax identical to the corresponding HTTP headers in HTTP/1.1 [7], except that the first entry of each list must be a definite (non-wildcarded) type specification. This first entry is used as default type information.

As attributes of linking elements (such as <EMBED>), these constraints support efficient type-negotiation, by narrowing the range of types in advance. For instance, a link with ACCEPT="video/mpeg, video/*" suggests that multiple video formats may be available for negotiation; if the the only video format that the user-agent can usefully interpret is video/avi, a reasonable HTTP negotiation offer would be Accept: video/avi, video/mpeg (with a video/mpeg result stored to disk).

In link specializer elements defining name-value pairs (<INPUT> and <PARAM>), these attributes specify the parameter types that should be acceptable to the method and resource in question (retrieval and presentation resource, resp.). When the user is allowed to change the types of the values (as is permitted by RFC 1867 [8] for <INPUT>s of TYPE=FILE in <FORM> links, for example), these attributes may therefore be used constrain the user's changes, as a service to the eventual data consumer. In addition, the ACCEPT attribute (but not ACCEPT-CHARSET or ACCEPT-ENCODING) should be used to interpret the default value provided in the markup (the VALUE attribute).

Note on MIME/SGML/I18N/FORM Interactions: Resources which rely on form submission (or presentation specializers) in an internationalized setting must be aware of one important caveat: just because the default value (the VALUE attribute) of a field is returned, it cannot be guaranteed to be returned as an identical sequence of octets -- only as (another) valid encoding of the same sequence of characters (under the character- and transfer-encodings declared for the field in the form submission).

This may be true even if the character- and transfer-encodings of the original HTML MIME entity containing the form are the same as the eventual encodings used for form submission. The reason is that

the MIME-to-SGML process is not required to preserve any of the encoding information of the MIME entity, only the actual sequence of characters in the (abstract) document character set of the SGML document entity. [This note should be moved to the HTML I18N draft.]

```
<![ %HTML.Recommended [
     <!ENTITY % key.value  -- key/value pair with typed value --
             'NAME    NAME       #IMPLIED   -- keyword, usually
required --
              VALUE   CDATA      #IMPLIED   -- default value --
              %mime.constraints;            -- type constraints
for value --'>
]]>
<!ENTITY % key.value  -- key/value pair with typed value --
     'NAME    CDATA      #IMPLIED   -- keyword, usually required --
      VALUE   CDATA      #IMPLIED   -- default value --
      %mime.constraints;            -- type constraints for
value --'>
```

NAME
A parameter name. It is not required that NAMEs be unique keys, even within the scope of a single linking element; a multiple-values model is supported. With the strict HTML DTD, this attribute is a case-insensitive SGML NAME token. With the

default HTML DTD, this attribute may be arbitrary CDATA, and any desired case-folding is the responsibility of the eventual consumer.

VALUE
Tags with %key.value attributes are used to collectively build up link specializers. Abstractly, the WWW method specializer resulting from such a collection of tags is the MIME entity of type multipart/form-data [8] which encodes all the name-value pairs and their types. Any %key.value tags without NAME attributes do not contribute to the specializer.

There are two exceptions to this general model of specializer construction. The first is historical: in the case of <INPUT> tags in a <FORM>, the default is to create a specializer of type

application/x-www-form-urlencoded (this can only unambiguously create a form submission with values having default MIME type). This historical behavior can be overridden by setting the <FORM>'s ENCTYPE attribute to "multipart/form-data" (which is strongly recommended). In case of legacy servers which require application/x-www-form-urlencoded form submission, but expect values of non-default MIME types, those expectations must be explicitly spelled out in the %mime.constraints. of each <INPUT>.

The second exception provides a way to build small method specializers of arbitrary character-based type; the rule is that if exactly one instance of a %key.value tag is supplied, and that tag does not have a NAME attribute, then its VALUE is the entire specializer entity.

```
   <!ENTITY % link.metainfo  -- overall link metainfo --
        'TITLE   CDATA      #IMPLIED   -- overall title for resource --
         URN     %URI       #IMPLIED   -- global entity name --
         REL     %linkType  #IMPLIED   -- link relationship --
         REV     %linkType  #IMPLIED   -- reverse relationship --'>
   <![ %HTML.Deprecated [
        <!ENTITY % case.metainfo  -- link metainfo for specific
variant --
                '%mime.constraints;             -- type constraints --
                 METHODS  NAMES     #IMPLIED    -- deprecated --'>
   ]]>
   <!ENTITY % case.metainfo  -- link metainfo for specific variant --
        '%mime.constraints;             -- type constraints --'>
   <!ENTITY % linkExtraAttributes
        '%link.metainfo;
         %case.metainfo;
        '>
```

The link meta-information attribute set %linkExtraAttributes has

been logically split into two parts to ease future extension. The new %case.metainfo component represents meta-information specific to a single link variant (currently consisting of MIME type information).

The only other change is the deprecation of the METHODS attribute. It is difficult to define the semantics of the METHODS attribute in an object-oriented model of the Web.

```
<!ENTITY % size
     'WIDTH   %Length    #IMPLIED   -- desired width in units --
      HEIGHT  %Length    #IMPLIED   -- desired height in units --'>
```

The %size; attribute set is used for elements that require bounding-box hints for rapid layout.

```
<!ENTITY % align.simple
     'ALIGN   (top|middle|bottom) #IMPLIED  -- glyph alignment --'>
<!ENTITY % align
     'ALIGN   (top|middle|bottom|left|center|right)  #IMPLIED
      HSPACE  %Length    #IMPLIED   -- text stays this far away
horizontally --
      VSPACE  %Length    #IMPLIED   -- text stays this far away
vertically --
      FLOWTO  NAME       #IMPLIED   -- flow text around block
until this ID --'>
```

The %align; attribute set is used for graphic elements that can function either as floats or glyphs, while %align.simple; is used for graphic elements that function only as glyphs.

ALIGN
: The value of this attribute indicates the relation between the graphic element and the text flow. The possible values are:

LEFT
RIGHT
: The graphic element functions as a float; subsequent text will be flowed around the graphic. The left (resp. right) edge of the graphic is aligned with the left (resp. right) edge of the text flow area.

CENTER
: The graphic element functions as a float centered horizontally within the text flow area. The graphic interrupts the flow of text, but does not logically break

the current paragraph.

TOP
BOTTOM
The graphic element functions as a glyph within the line of text. The top (resp. bottom) of the graphic is aligned with the highest point (resp. baseline) of the rendered line of text.

MIDDLE
The graphic element functions as a glyph within the line of text. The baseline of the graphic is aligned with the baseline of the rendered line of text. By default, the baseline of the image is taken to be at its vertical center. (There is currently no way to override this default baseline.)

HSPACE
VSPACE
The text surrounding the graphic element remains at least this far away. HSPACE specifies the minimum horizontal distance between the text and the left/right edges of the graphic, while VSPACE specifies the minimum vertical distance between the text and its top/bottom edges. This padding does not offset the alignment in any way.

FLOWTO
Sometimes it is important that some subsequent element appear below a float, rather than as part of the text flow alongside it; the ID of that element can be specified with the FLOWTO attribute.

This attribute provides a more structured alternative to <BR CLEAR=ALL>. Its purpose is to declare the range of text with which the float is associated (from the element itself, until the element whose ID is given by FLOWTO). In typical rendering, the text beyond the specified range would not be flowed around the graphic.

6. Summary of DTD Changes for HTML

The resulting SGML doctype is provisionally called "-//IETF//DTD HTML 2.1E//EN", and is defined in terms of the doctype "-//IETF//DTD

HTML 2.1//EN". Because the HTML 2.1 DTD [4] defines multibyte entity references, it must be used with an SGML declaration that provides an extended coded character set; the HTML 2.1E DTD inherits this restriction.

6.1. Retroactive Modifications for HTML 2.1

The following changes are included below, but we recommend that they be included directly into future drafts of HTML 2.1, for consistency. This section will be removed from future versions of

this draft, once these issues are resolved.

CHARSET in %linkExtraAttributes
: Change to ACCEPT-CHARSET to allow content negotiation.

ACCEPT-CHARSET in <FORM>
: Echoing comments of Larry Masinter, drop for now. This is wrong because it conflicts with media type metainfo for the FORM link, which should have priority since client Accept-ance was invented first! An ACCEPT-CHARSET should be given to each relevant INPUT instead.

ACCEPT in <INPUT>
: Add ACCEPT-CHARSET and ACCEPT-ENCODING here for constraining input values (not in <FORM> element).

METHODS in %linkExtraAttributes
: This should eventually be replaced by a per-link-variant METHOD attribute; multiple methods may have conflicting type signatures, not to mention semantics and security implications.

6.2. Incremental DTD for HTML 2.1E

The proposed HTML 2.1E DTD may be constructed unambiguously from the HTML 2.1 DTD [4] and the DTD fragment below by the following procedure: Entity and element definitions already in HTML 2.1 are overridden (respecting the marked sections); all others are added to the end of the DTD in the order they appear here.

```
<!-- This DTD fragment shows only changes from HTML 2.1 to 2.1E -->
```

```
   <!ENTITY % HTML.Version "-//IETF//DTD HTML 2.1E//EN">

   <!-- Content models -->

   <![ %HTML.Highlighting [
   <!ENTITY % text
"#PCDATA|A|EMBED|IMG|BR|%phrase|%font|SPAN|Q|BDO|SUP|SUB">
   ]]>
   <!ENTITY % text "#PCDATA|A|EMBED|IMG|BR|SPAN|Q|BDO|SUP|SUB">

   <!-- Attribute value types -->

   <!ENTITY % URI       "CDATA"    -- uniform resource identifier -->
   <!ENTITY % Length    "CDATA"    -- number followed by optional
units -->

   <!-- Attribute sets -->

   <!ENTITY % attrs
        'ID      ID         #IMPLIED   -- element identifier --
         LANG    NAME       #IMPLIED   -- RFC 1766 language tag --
```



```
         DIR     (ltr|rtl)  #IMPLIED   -- text directionality --'>
   <!ENTITY % size
        'WIDTH   %Length    #IMPLIED   -- desired width in units --
         HEIGHT  %Length    #IMPLIED   -- desired height in units --'>
   <!ENTITY % align.simple
        'ALIGN   (top|middle|bottom) #IMPLIED  -- glyph alignment --'>
   <!ENTITY % align
        'ALIGN   (top|middle|bottom|left|center|right)  #IMPLIED
         HSPACE  %Length    #IMPLIED   -- text stays this far away
horizontally --
         VSPACE  %Length    #IMPLIED   -- text stays this far away
vertically --
         FLOWTO  NAME       #IMPLIED   -- flow text around block
until this ID --'>
   <!ENTITY % mime.constraints -- MIME typing constraints --
        'ACCEPT           CDATA  "text/plain" -- applicable media
type(s) --
         ACCEPT-CHARSET   CDATA  "ISO-8859-1" -- appl. character
encoding(s) --
         ACCEPT-ENCODING  CDATA  "ISO-8859-1" -- appl. transfer
encoding(s) --'>
```

```
   <!ENTITY % link.internal -- links with implicit presentation
resource --
        'SRC     %URI;      #REQUIRED  -- resource to retrieve --
         PARAMS  CDATA      #IMPLIED   -- presentation specializer --
         USEMAP  %URI       #IMPLIED   -- default event handling,
if needed --'>
   <!ENTITY % link.metainfo  -- overall link metainfo  --
        'TITLE   CDATA      #IMPLIED   -- overall title for resource --
         URN     %URI       #IMPLIED   -- global entity name --
         REL     %linkType  #IMPLIED   -- link relationship --
         REV     %linkType  #IMPLIED   -- reverse relationship --'>
   <![ %HTML.Deprecated [
        <!ENTITY % case.metainfo  -- link metainfo for specific
variant --
                '%mime.constraints;             -- type constraints --
                 METHODS  NAMES     #IMPLIED    -- deprecated --'>
   ]]>
   <!ENTITY % case.metainfo  -- link metainfo for specific variant --
        '%mime.constraints;             -- type constraints --'>
   <!ENTITY % linkExtraAttributes
        '%link.metainfo;
         %case.metainfo;
        '>
   <![ %HTML.Recommended [
        <!ENTITY % key.value  -- key/value pair with typed value --
                'NAME    NAME       #IMPLIED   -- keyword, usually
required --
                 VALUE   CDATA      #IMPLIED   -- default value --
                 %mime.constraints;            -- type constraints
for value --'>
   ]]>
   <!ENTITY % key.value  -- key/value pair with typed value --
        'NAME    CDATA      #IMPLIED   -- keyword, usually required --
         VALUE   CDATA      #IMPLIED   -- default value --
         %mime.constraints;            -- type constraints for
value --'>

   <!-- Embed and Related Elements -->

   <!ELEMENT EMBED - - (PARAM*, CAPTION?, EMBED.BODY, CREDIT?)>
   <!ATTLIST EMBED
        %attrs;
        %link.internal;
        %link.metainfo;
```

```
        %case.metainfo;
        %size;                      -- size of reserved area --
        %align;                     -- alignment or float --
        %SDAPREF; '<Fig><Xref IDRef="#AttVal(SRC)"><?SDATrans
Embed: #AttList>'
        %SDASUFF; '</Fig>'
        >
   <!ELEMENT CAPTION - - (%text;)+ -- caption for floating element -->
   <!ATTLIST CAPTION
        %attrs;
        %align;                     -- side of rectangle where
caption placed --
        %SDAPREF; "Caption: "
        >
   <!ELEMENT EMBED.BODY O O %A.content>
   <!ATTLIST EMBED.BODY
        %attrs;
        >
   <!ELEMENT CREDIT - - (%text;)* -- copyright/credit for embedded
object  -->
   <!ATTLIST CREDIT
        %attrs;
        %SDAFORM; "Fn"
        >

   <!-- Link Specializer Elements -->

   <!ELEMENT INPUT - O EMPTY  -- builds retrieval specializer -- >
   <!ATTLIST INPUT
        %attrs;
        %key.value;                      -- key and MIME-typed value --
        TYPE       %InputType  #IMPLIED  -- defaults to TEXT in
FORM context --
        CHECKED    (CHECKED)   #IMPLIED  -- initial boolean state --
        SRC        %URI;       #IMPLIED  -- embedded graphic for
TYPE=IMAGE --
        %align.simple;                   -- alignment for TYPE=IMAGE --
        SIZE       CDATA       #IMPLIED
        MAXLENGTH  NUMBER      #IMPLIED
        %SDAPREF; "Input #AttVal(Type): "
        >
   <!ELEMENT PARAM - O EMPTY  -- builds presentation specializer -- >
   <!ATTLIST PARAM
        %attrs;
        %key.value;
        %SDAPREF; "<?SDATrans Param: #AttList>"
        >

   <!-- Corrections -->
```

```
<!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
<!ATTLIST FORM
     %attrs;
     ACTION CDATA #IMPLIED
     METHOD (%HTTP-Method) GET
     ENCTYPE %Content-Type; "application/x-www-form-urlencoded"
     %SDAPREF; "<Para>Form:</Para>"
     %SDASUFF; "<Para>Form End.</Para>"
```



```
     >
```

7. Transition Issues

7.1. Sun's APPLET Element

Sun's <APPLET> element [9] can be mapped to <EMBED> with essentially just a few name changes:

* APPLET becomes EMBED

* NAME becomes ID

* CODE is prefixed with CODEBASE to become SRC

* CODEBASE is not needed (the embedding API is required to pass the retrieval URI (the SRC) to the media handler, from which the CODEBASE can be deduced)

It is possible to construct peculiar examples where the CODEBASE is not the base URI of the combination of CODEBASE with CODE, but this seems more confusing than useful.

7.2. Netscape's Early Implementations of EMBED

Netscape initially implemented <EMBED> as an empty element [10], not a container. While this causes some forward-compatibility problems, the %A.content content model of <EMBED.BODY> allows error recovery to occur after the first paragraph break when </EMBED> is omitted.

Netscape's initial implementation also uses arbitrary attributes to pass parameters. In the interim, authors can use a combination like the following for compatibility (though it's still not legal SGML):

```
<EMBED SRC="sample.app" ALT="simple alt text" FOO=3 BAR=9>
        <PARAM NAME=foo VALUE=3> <PARAM NAME=bar VALUE=9>
</EMBED>
```

8. Security Considerations

Immediate invocation of link without user feedback or control aggravates security problems of links described in [11].

9. Acknowledgments

Dave Raggett did a lot of the early work on <FIG> [12] which is reflected here in <EMBED>.

Arthur van Hoff of Sun Microsystems designed <APPLET> based on discussion in the hotjava-interest mailing list. This proposal has

been strongly influenced by the <APPLET> tag.

Dan Connolly began work to formalize the Web model [13]. Discussions with Larry Masinter and Gavin Nicol were also important in shaping the model presented here.

10. References

1. T. Berners-Lee and D. Connolly, HTML 2.0, RFC 1866.

2. J. Postel, Internet Media Types, RFC 1590.

3. R. Fielding, Relative Uniform Resource Locators, RFC 1808.

4. F. Yergeau, G. Nicol, G. Adams, and M. Duerst, HTML Internationalization.

5. N. Borenstein and N. Freed, MIME Content Types, RFC 1521.

6. T. Berners-Lee, Universal Resource Identifiers, RFC 1630.

7. T. Berners-Lee, R. Fielding, and H. F. Nielsen, HTTP/1.1.

8. Larry Masinter et al., Form-Based File Upload in HTML, RFC 1867.

9. Sun Microsystems, APPLET Element Syntax.

10. Netscape Communications, Netscape Navigator 2.0 Feature Descriptions

11. T. Berners-Lee, L. Masinter, and M. McCahill, Uniform Resource Locators, RFC 1738.

12. Dave Raggett, HTML3 Draft.

13. Daniel W. Connolly, Resource Discovery and Reliable Links.

11. Authors' Addresses

Paul Burchard
<burchard@cs.princeton.edu>
Computer Science Dept.
Princeton University
35 Olden Street
Princeton NJ 08544

Dave Raggett
<dsr@w3.org>
World Wide Web Consortium

Massachusetts Institute of Technology
545 Technology Square
Cambridge MA 02139