MIMESGML Working Group D. Stinchfield
INTERNET-DRAFT EBT, Inc.
<draft-ietf- ?????? >
Expires February 1, 1996 August 31, 1995

Using Catalogs and MIME to Exchange SGML Documents

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the HTTP working group at <sgml-internet@ebt.com>. Discussions of the working group are archived at ftp://ftp.naggum.no/pub/SGML-internet.

Abstract

This draft proposes a standard for exchanging SGML documents over the World Wide Web using catalogs and MIME. This draft extends SGML Open's definition of catalogs [10] by adding to it new keywords and storage object identifier (SOI) types. The new keywords identify SGML document objects (such as document type declarations and document entities) and non-SGML document objects (such as stylesheets). The new SOI types include URIs and MIME Content-IDs. This document also defines two new MIME content types called Application/SGML-Catalog and Application/SGML. Application/SGML-Catalog identifies a MIME body part as a catalog, while Application/SGML identifies a MIME body part as an SGML object.

1. Introduction

This draft proposes a standard for exchanging SGML documents over the World Wide Web using catalogs and MIME. This draft extends SGML Open's definition of catalogs [10] by adding to it new keywords and storage object identifier (SOI) types. The new keywords identify SGML document objects (such as document type declarations and document entities) and non-SGML document objects (such as stylesheets). The new SOI types include URIs and MIME Content-IDs. This document also defines two new MIME content types called Application/SGML-Catalog and Application/SGML. Application/SGML-Catalog identifies a MIME body part as a catalog, while Application/SGML identifies a MIME body part as an SGML object.

SGML catalogs (referred to as "catalogs" and as "export catalogs" in this document) were introduced in the SGML Open document entitled "Entity Management" [10] (also referred to as TR9401). Catalog entries, as described in the SGML Open document, provide a mapping of PUBLIC external identifiers and entity names to system-dependent SOIs (these system-dependent SOIs are typically filenames, see section 2.2 for details). This system-dependent requirement is too restrictive and fails to meet the needs of the internet community. Specifically, SGML Open's catalog definition does not provide keyword entries for all the types of external objects used in SGML and it does not define how to map catalog entries to system-independent SOIs such as URLs. This document addresses both of these limitations by extending SGML Open's catalog definition to include keywords for all possible SGML external identifiers and by defining system-independent SOIs.

Some key benefits to using catalogs and MIME to exchange SGML documents are:

- o a client only needs a catalog to begin processing, it simply fetches the components referenced in the catalog as they are needed;
- o a client that understands catalogs has a way to fetch components of a document that it doesn't already have;
- o document components do not have to be modified in order to be referenced in a catalog;
- o components of a document can be distributed across many servers;
- o catalogs do not depend on MIME, therefore, they can be used in other packaging schemes;
- o the impact on MIME is minimized;
- o catalogs are an implemented, proven technology;
- o a document's system identifiers can be referenced in a catalog and subsequently resolved by a client.

1.1 Overview

The new keywords for SGML document components have been derived from Charles Goldfarb's paper entitled "Entity Management in SGML" [11]. These new keywords are DOCTYPE, NOTATION, BASESET, CAPACITY, DOCENTITY, and SGMLDECL. The last two, DOCENTITY and SGMLDECL, are descriptive of the terms document entity and SGML declaration that are referred to in Goldfarb's paper as the "undeclared entity" and as "implied SGML".

Two other keywords are defined in this document that, unlike the previously mentioned keywords, do not identify SGML objects. They are called BASEURL and SEMANTICS. BASEURL is used to resolve relative URLs found in a catalog. SEMANTICS are used to reference semantic processing information like stylesheets.

User defined keywords are allowed in catalogs for experimental purposes. User defined keywords must begin with "X-" or "x-" and must be located at the end of the catalog - this makes it easy for clients to identify user defined keywords and to ignore them if they wish.

System-independent SOIs are defined to permit both URIs and MIME Content-IDs. The usefulness of URIs is evident from the popularity of the World Wide Web refer to [7] [8] [12] and [4] for detailed descriptions of URIs, URNs, and URLs. An SOI that is defined to be a MIME Content-ID identifies a document component that is contained in a MIME body part. Refer to [13] for a description of MIME Content-IDs.

There are a number of ways to serve up an SGML document using catalogs and MIME. The server could deliver just a catalog; the server could deliver a catalog and a document entity; the server could deliver a catalog and all of the document's components; the server could deliver a catalog and a non-document entity - detailed examples of all of these can be found in section 5. Any catalog served using MIME will have a Content-Type of Application/SGML-Catalog (see section 5.1 for detailed examples).

1.2 SGML document Components

This section describes the components of an SGML document. This is not meant to be a rigorous description of SGML, for that the reader should refer to ISO 8879:1986 [14] and to Charles Goldfarb's "The SGML Handbook" [3].

SGML documents are typically made up of a number of components. Some of the components contain instructions to the SGML parser (such as the SGML declaration and the DTD) while others contain marked up text (such as the SGML document entity), and still others contain non-SGML data (such as figures). These components can be identified in a number of different ways using what SGML calls external identifiers. Catalog entries identify SGML document components.

1.3 Terminology

Will be added shortly.

2. Catalog Description

A catalog provides a mapping between external object identifiers (such as public identifiers and entity names) to SOIs. In TR9401 [10] an SGML system's entity manager typically treats SOIs as system-dependent filenames. This is too restrictive for SGML Systems that need to take advantage of the Internet. Formally extending the meaning of SOIs to include Universal Resource Identifiers (URI, [7]) removes this system-dependent restriction. Catalogs that contain these types of SOIs are system-independent and so give flexibility to SGML systems.

To interchange an SGML document means to send a catalog and a set of zero or more components to a client (note, the interchange package could contain all of the document's components). The catalog contains references to document components (refer to section 2.1 for a detailed list of keywords).

Entity references that are declared with system identifiers can be referenced in a catalog. To process these types of catalog entries the client SGML parser uses TR9401's [10] catalog processing mode 2 to resolve entity references (TR9401 describes 2 modes of catalog access: the first mode tells the application to use system identifiers found in the document as the SOI; the second mode tells the application to use the catalog entry as the SOI for the entity reference, even when there is a system identifier declared for it).

2.1 Catalog Keywords

A catalog contains entries for SGML External Identifiers, for Semantic Information Identifiers, and for a base URL. The order of the entries in the catalog is not important. All entries are optional. A catalog can contain multiple entries with the same keyword. The following keywords are defined in this document:

```
SGMLDECL - SGML declaration
BASESET - base character set (part of the SGML declaration)
CAPACITY - capacity set (part of the SGML declaration)
SYNTAX - concrete syntax (part of the SGML declaration)
DOCENTITY - SGML document entity
DOCTYPE - Document type declaration (DTD)
PUBLIC - public external identifier
ENTITY - entity name
NOTATION - notation name
SEMANTICS - name and type of the semantic information
BASEURL - base URL
```

These keywords are necessary to keep the name spaces of an SGML document separate. An SGML document may contain many components, each of which can be identified by a local name (such as "chap1") or by a global "public identifier" (such as ISBN, URN, etc.). Local identifiers may be re-used for different kinds of components. For example, an SGML document could have an included entity called "footnote" and a special notation called "footnote". In the catalog these would be represented by the following entries:

```
ENTITY   "footnote"  "http://blah.com/blah.foot"
NOTATION "footnote "http://blah.com/notation/yuk.not"
```

2.1.1 SGMLDECL, BASESET, CAPACITY, SYNTAX

The SGML declaration is part of the SGML document and is required by the SGML parser before it can begin parsing. The SGML declaration defines, among many other things, the SGML document's character set and the character strings that define the markup. For an excellent description of SGML declarations refer to Wayne Wohler's paper called "SGML declarations" [1]. Appendix A of this document contains an example of an SGML declaration. If an SGML document is not explicitly associated with an SGML declaration then a default SGML declaration is assumed by the parser.

The SGML declaration is identified in the catalog by the SGMLDECL keyword. The SGML declaration is not always self-contained. It can include references to public identifiers ([3], 378). The following parameters of an SGML declaration can be defined as public identifiers: BASESET ([3], 453:12); CAPACITY ([3], 456:2); and SYNTAX ([3], 458:2).

The catalog keywords for the SGML declaration and for the above parameters have the following syntax:

```
   sgmldecl = ("SGMLDECL",
     ps+, storage object identifier)
   baseset  = ("BASESET",
     ps+, public identifier, ps+, storage object identifier)
   capacity = ("CAPACITY",
     ps+, public identifier, ps+, storage object identifier)
   syntax   = ("SYNTAX",
     ps+, public identifier, ps+, storage object identifier)
```

Note, the SGML declaration is part of the document it describes. As such it must be encoded in the document's character set. The client must be told the character encoding of the SGML declaration before any processing can begin. The "charset" parameter of a MIME body part that contains the SGML declaration identifies its character set.

2.1.2 DOCENTITY

The DOCENTITY catalog keyword refers to the SGML document entity ([3], 142:1). The SGML document entity describes the first entity of the SGML document ([3], 142:1). It typically contains a reference to a DTD, a document type declaration subset ([3], 404:6), and marked-up text. Appendix C of this document contains an example of an SGML document entity. The syntax for DOCENTITY follows:

```
     docentity   = ("DOCENTITY",  ps+, storage object identifier)
```

2.1.3 DOCTYPE

The DOCTYPE catalog keyword refers to the Document type declaration (DTD) ([3], p402). The syntax for DOCTYPE is:

```
     doctype  = ("DOCTYPE",
        ps+, document type name, ps+, storage object identifier)
```

The value of the "document type name" is defined in the SGML document entity, for example the first line in the Document entity used in the examples looks like this:

```
     <!DOCTYPE MEMO  "/usr/wcs/dtd/memo.dtd" >
```

A catalog entry for it would look something like this:

```
     DOCTYPE  "MEMO"       "../../dtds/memo.dtd"
```

Note how the name, "MEMO", is defined in the document and referenced in the catalog. The entity manager resolves a reference to the DTD by looking its name up in the catalog. Appendix B of this document contains an example of a DTD.

It is not uncommon for the document’s DTD to have both a public identifier and a system identifier:

```
     <!DOCTYPE MEMO  PUBLIC "-//EBT//DTD Released Memo//EN"
                         "/usr/wcs/dtds/memo.dtd" [...] > ...
```

For these cases the server decides whether or not to include both in the catalog. If it decides to include both then the catalog entries for them would look something like this:

```
     DOCTYPE  "MEMO"  "http:/www.bill.com/usr/wcs/dtds/memo.dtd"
     PUBLIC   "-//EBT//DTD Released Memo//EN"
                      "http://www.yoman.edu/pub/dtds/memo.dtd"
```

The client decides which definition to use first.

2.1.4 PUBLIC

The PUBLIC catalog keyword refers to public identifiers that are defined in the DTD or in the Document entity’s "document type declaration subset". The syntax for the PUBLIC keyword is:

```
     public = ("PUBLIC",
        ps+, public identifier, ps+, storage object identifier)
```

Here's an example of a parameter entity declaration that contains a public identifier:

```
<!ENTITY % ISOnum PUBLIC
           "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN">
```

a catalog entry for the above public identifier might look like this:

```
PUBLIC  "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN"
        "http://www.wcs.com/usr/wcs/isonum.ent"
```

Note, the public identifiers defined in the SGML declaration need not be referenced in the catalog with the PUBLIC keyword. These keywords are used instead (see 2.1.2): BASESET, CAPACITY, and SYNTAX.

2.1.5 ENTITY

The ENTITY catalog keyword refers to document entities that are defined/referenced in the document but have no public identifiers. The syntax for the ENTITY keyword is:

```
entity = ("ENTITY",
   ps+, entity name, ps+, storage object identifier)
```

The following are examples of entity declarations for system identifiers:

```
<!ENTITY  Legal    SYSTEM  >
           and
<!ENTITY  MyEnding SYSTEM "/usr/des/ending.sgm"  >
```

Entries in a catalog for these would look something like this:

```
ENTITY   "Legal"      "http://www.bill.com/company/legal.sgm"
ENTITY   "MyEnding"   "http://www.bill.com/ending.sgml"
```

2.1.6 NOTATION

The NOTATION catalog keyword refers to data content notations defined/referenced in the document. The syntax for NOTATION is:

```
notation = ("NOTATION",
   ps+, notation name, ps+, storage object identifier)
```

Examples of notation declarations are:

```
<!NOTATION  TCL  SYSTEM >
         and
<!NOTATION  TeX  SYSTEM "c:\eqn\bin\eqn.exe">
```

Entries in a catalog for these would look something like this:

```
NOTATION  "TCL"  "http://www.bill.com/notation/tcl"
NOTATION  "TeX"  "http://www.bill.com/notation/eqn.exe"
```

2.1.7 SEMANTICS

There may be semantic information, such as stylesheets, associated with a document. Semantic information is not required to parse the document and can be ignored by the client. However, it is often required that a client be able to access appropriate semantic specifications. The syntax for the SEMANTICS keyword is:

```
semantics = ("SEMANTICS",
   ps+, semantic name, ps+, semantic type, ps+,
   storage object identifier)
```

Here’s an example of an entry in a catalog for semantic information:

```
SEMANTICS "large-print"  "DSSSL"
   "http://www.bill.com/style/large.sty"
```

2.1.8 BASEURL

Relative URLs [4] are allowed in SOIs. Relative URLs can be resolved using the BASEURL keyword catalog entry. If there's no BASEURL in the catalog then the URL for the catalog is used for relative URL resolution. The syntax for the BASEURL keyword is:

```
baseurl = ("BASEURL", ps+, absoluteURL)
```

Here's an example of an entry in a catalog for BASEURL:

```
BASEURL  "http://www.bill.com/docs/memo/mine/dummy"
```

2.1.9 User Defined Keywords

A user can create new catalog keywords by beginning the keyword with either an "X-" or an "x-". Users may do this to test experimental keywords. User defined keywords are only allowed at the end of the catalog and must begin with either the "X-" or the "x-" prefix.

2.2 Storage Object Identifiers

As described in TR9401 [10], an SOI "is expected to be a string that is assumed to make sense to the operating system involved, i.e., it should name a file accessible from the current file system" ([10], p. 4, Notes: b). TR9401 anticipated the extension of SOIs to define "a different or extended meaning that will require the recognition and special processing of certain characters in the SOI." ([10], p.5). Two such SOI extensions are defined in this section. The first defines SOIs in terms of URIs and the second defines them in terms of a MIME Content-ID. The latter type of SOI is used to identify the body part of a Multipart/Related message. Content-ID referencing is used when a single Multipart/Related message contains the document's catalog and one or more of the document's components (see 5.2 and 5.3 for examples). The syntax for an SOI is:

```
storage object identifier =
       uri object identifier             |
       content id object identifier      |
       TR9401 storage object identifier
```

The term "TR9401 storage object identifier" refers to the TR9401's definition of an SOI and is included here for backwards compatibility. The URL "file:" scheme could also be used to represent filenames.

2.2.1 URIs as SOIs

URIs are used to describe the names and locations of objects. Uniform Resource Locators (URL) and Uniform Resource Names (URN) are examples of URIs (refer to [8] and [12] for rigorous descriptions).

A URL defines a location of an object that can be accessed typically via the internet.   Generally speaking URLs have the following structure:

```
      scheme:scheme-specific-part
```

A scheme is associated with a protocol such as http or ftp. The scheme-specific-part contains the information required by the scheme to locate the object. There are a number of other schemes in addition to the ones mentioned here (see [8]).

"The purpose or function of a URN is to provide a globally unique, persistent identifier used for recognition, for access to characteristics of the resource or for access to the resource itself." [12] There is no internet standard defined for URNs yet, but one is anticipated soon.

2.2.2 The Content-ID SOI

In addition to the catalog a server can send some/all of the document's components in a MIME message. Content-ID SOIs are used to map entries in the catalog to the body part of the MIME message that contains the corresponding component. See the examples in 5.2 and 5.3. The syntax for Content-ID based SOIs is:

```
     content id object identifier = "Content-ID" ":" content id

     content id = as defined in RFC 1521
```

3. Export Catalog Syntax

```
export catalog =
   ( catalog entry, ps+ )+, ( user defined, ps+ )*

catalog entry  =
          sgmldecl    |
          baseset     |
          capacity    |
          syntax      |
          docentity   |
          doctype     |
          public      |
          entity      |
          notation    |
          semantics   |
          baseurl

sgmldecl =
   ("SGMLDECL", ps+, storage object identifier)

baseset  =
   ("BASESET",  ps+, public identifier, ps+, storage object identifier)

capacity =
   ("CAPACITY", ps+, public identifier, ps+, storage object identifier)

syntax   =
```

```
   ("SYNTAX",   ps+, public identifier, ps+, storage object identifier)

docentity=
   ("DOCENTITY",ps+, storage object identifier)

doctype  =
   ("DOCTYPE",  ps+, document type name, ps+, storage object identifier)

public   =
   ("PUBLIC",   ps+, public identifier,  ps+, storage object identifier)

entity   =
   ("ENTITY",   ps+, entity name, ps+, storage object identifier)

notation =
   ("NOTATION", ps+, notation name, ps+, storage object identifier)

semantics=
   ("SEMANTICS",ps+, semantic name, ps+, semantic type,
                                          storage object identifier)
baseurl  =
   ("BASEURL",  ps+, absoluteURL)

user defined = (("X-" | "x-"), alphanumeric+)

storage object identifier =
          uri object identifier        |
          content id object identifier |
          TR9401 storage object identifier

uri object identifier =
          as defined in RFCs 1808[4], 1630[7], 1738[8]

content id object identifier =
          "Content-ID" ":" content id

content id = as defined in RFC 1521

TR9401 storage object identifier =
          "storage object identifier" as defined in TR9401 [10]

semantic name = alphanumeric+

absoluteURL = see "absoluteURL" in RFC 1808 [4]

document type name = alphanumeric+

; From TR9401 [10]
ps = s | comment

LIT  = '"'      ; the double quote

LITA = "'"      ; the single quote

comment = COM, system character*, COM

COM  = "--"

entity name   =
```

```
          extended name character+               |
          (LIT, extended name character+, LIT)  |
          (LITA, extended name character+, LITA)
```

The following notes are taken from TR9401 [10]:

(1.) public identifier and s are defined in 8879 (and RS, RE, SPACE and SEPCHAR are as in the reference concrete syntax of 8879);

(2.) extended name character means (a) in the case of an undelimited string, any character except the "null" character, the LIT character, the LITA character, and those characters allowed in s, and (b) in the case of a delimited literal, any character except the "null" character and the delimiting character for that literal (i.e., LIT or LITA);

(3.) system character means (a) in the case of an undelimited string, any character except the "null" character, the LIT character, the LITA character, and those characters in s; (b) in the case of delimited literal, any character except the "null" character and the delimiting character for the literal (i.e., LIT or LITA); (c) in the case of a comment, any character except the "null" character and a sequence of characters that would be interpreted as the terminating COM delimiter; (d) in the case of an undelimited string the comprises the keyword and the second and subsequent argument of other information, the string must not be recognizable as the PUBLIC or ENTITY keywords.

4. Using MIME

Two new MIME Content-Types called Application/SGML-Catalog and Application/SGML are defined in this draft. Application/SGML-Catalog identifies a MIME body part for a catalog and Application/SGML identifies a MIME body part for an SGML document component (also referred to as an SGML object in this document).

The MIME Multipart/Related content-type [5] is a useful way to package up a catalog and one or more document components into a single MIME message. The examples (section 5) make extensive use of Multipart/Related. However, there is nothing to prevent a server from using some other content type besides Multipart/Related for encapsulation or from doing no encapsulation at all.

5. Examples

The SGML document used in all the examples is composed of the following components:

- o an SGML declaration, defined in Appendix A;
- o a Document type declaration (DTD), defined in Appendix B;
- o an SGML document entity, defined in Appendix C;
- o two SGML entities, defined in Appendix C;
- o a figure entity, not defined in this draft.

In all examples the components of this SGML document are spread across multiple servers, except for the example entitled "Sending a Catalog and All of its Components", where all of the document’s are contained in a single MIME message.

Each example defines its own unique catalog. The variation of the catalogs from example to example is slight and depends on the number of document components that are being sent along with the catalog. Remember, zero or more components can be sent with the catalog. The sender decides how many components to include in the MIME message. The recipient (client that is) has enough information to then obtain any other component when they are needed.

A document component that's not included in the MIME message can be resolved by the client in one of two ways: 1.) the client has the component cached; 2.) the client requests the component using the SOI defined for it in the catalog.

The definitions for the following external identifiers are not included in this document:

```
     formal public identifiers
          ISO 646:1983//
              CHARSET International Reference Version (IRV)//
              ESC 2/5 4/0
          ISO 8879-1986//
              ENTITIES Numeric and Special Graphic//EN

      system identifier
           ../style/all.sty" - DSSSL style sheet
```

Examples that contain a Multipart/Related MIME Content Type default the compound object's "root" to the first body part of the message [5] which is always of type Application/SGML-Catalog.

5.1 Sending Only A Catalog

In this example only the catalog is sent to the client. If the client's SGML System is capable of handling URIs that are defined as SOIs then the newly received catalog can be passed, without modification, to the client's SGML System. If the client's SGML System cannot handle this type of SOI then the client must do some pre-processing before passing the catalog on. The pre-processing logic should do something like the following:

1. fetch all of the components referenced in the catalog,
2. store the components locally, and
3. update the catalog (change all URI based SOIs).

A server may have a number of reasons why it would want/need to send only a catalog:

- o The server only stores catalogs, it does not store any document components;
- o The client may have requested only the catalog. Perhaps the client wants to compare the contents of this catalog with the contents of a different catalog. Or maybe the client already has most, if not all, of the document's components cached;
- o The server may want to keep network traffic down by increasing the likelihood that the client will get a cache hit on catalog entries.

5.1.1 MIME Message Content

```
MIME-Version: 1.0
Content-Type: Application/SGML-Catalog; charset=us-ascii

SGMLDECL "http://www.ebt.com/decl/ebtsgml.dcl"
BASESET  "ISO 646:1983//CHARSET
         International Reference Version (IRV)//ESC 2/5 4/0"
         "http://www.iso.ch/charset/6461983.cha"
BASESET  "ISO Registration Number 100//CHARSET
         ECMA-94 Right-hand Part of Latin Alphabet Nr.1//ESC 2/13 4/1"
         "http://www.iso.ch/charset/ecma94.cha"
CAPACITY "-//EBT//CAPACITY CoolCaps 1.0//"
         "http://www.ebt.com/decl/coolcaps.cap"
SYNTAX   "-//EBT//SYNTAX  SinSyn 0.1//"
         "http://www.ebt.com/decl/syntax/sinsyn.syn"
BASEURL  "http://www.bill.com/docs/memo/mine/dummy"
DOCENTITY  "anaxi.sgm"
DOCTYPE  "MEMO"       "../../dtds/memo.dtd"
PUBLIC   "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN"
         "http://www.wcs.com/usr/wcs/isonum.ent"
ENTITY   "%ISOnum"    "http://www.wcs.com/usr/wcs/isonum.ent"
ENTITY   "MyEnding"   "ending.sgml"
ENTITY   "Legal"      "../company/legal.sgm"
SEMANTICS "large-print"  "DSSSL"   "../style/all.sty"
```

5.2 Sending a Catalog and the Document Entity

This example describes how to send a catalog and a document entity component using a Multipart/Related message [5]. This example is the likely scenario for Web-based Browsers where simultaneous rendering and resolving of external identifiers are necessary. The document entity will likely contain enough text for the Browser to render meaningful text to the user, but it won’t include the many entities that the text may link to. These external identifiers, like figures, can be resolved (fetched, that is) by the entity manager while the application is rendering the text or (as for hyperlinked information) on user demand.

5.2.1 MIME Message Content

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=let-go-of-my-leg;
              type="Application/SGML-Catalog"

--let-go-of-my-leg
Content-Type: Application/SGML-Catalog; charset=us-ascii

SGMLDECL "http://www.ebt.com/decl/ebtsgml.dcl"
BASESET  "ISO 646:1983//CHARSET
         International Reference Version (IRV)//ESC 2/5 4/0"
         "http://www.iso.ch/charset/6461983.cha"
BASESET  "ISO Registration Number 100//CHARSET
         ECMA-94 Right-hand Part of Latin Alphabet Nr.1//ESC 2/13 4/1"
```

```
         "http://www.iso.ch/charset/ecma94.cha"
CAPACITY "-//EBT//CAPACITY CoolCaps 1.0//"
         "http://www.ebt.com/decl/coolcaps.cap"
SYNTAX   "-//EBT//SYNTAX  SinSyn 0.1//"
         "http://www.ebt.com/decl/syntax/sinsyn.syn"
BASEURL  "http://www.bill.com/docs/memo/mine/dummy"
DOCENTITY  "Content-ID:<bookemdano>"
DOCTYPE  "MEMO"       "../../dtds/memo.dtd"
PUBLIC   "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN"
         "http://www.wcs.com/usr/wcs/isonum.ent"
ENTITY   "%ISOnum"    "http://www.wcs.com/usr/wcs/isonum.ent"
ENTITY   "MyEnding"   "ending.sgml"
ENTITY   "Legal"      "../company/legal.sgm"
SEMANTICS "large-print"  "DSSSL"   "../style/all.sty"

--let-go-of-my-leg
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <bookemdano>

include document entity from Appendix C
--let-go-of-my-leg--
```

5.3 Sending a Catalog and All Document Components

Like the previous example, sending a catalog and all of the document’s components is described using a Multipart/Related message. A server might do something like this in response to a client’s request for all of the document components to be sent with the catalog.

5.3.1 MIME Message Content

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=go-speed-racer;
              type="Application/SGML-Catalog"

--go-speed-racer
Content-Type: Application/SGML-Catalog; charset=us-ascii

SGMLDECL "Content-ID:<joejitsu>"
BASESET  "ISO 646:1983//CHARSET
         International Reference Version (IRV)//ESC 2/5 4/0"
         "Content-ID:<mumbles>"
BASESET  "ISO Registration Number 100//CHARSET
         ECMA-94 Right-hand Part of Latin Alphabet Nr.1//ESC 2/13 4/1"
         "Content-ID:<underdog>"
CAPACITY "-//EBT//CAPACITY CoolCaps 1.0//"
         "Content-ID:<mastercylinder>"
SYNTAX   "-//EBT//SYNTAX  SinSyn 0.1//"  "Content-ID:<lippythelion>"
DOCENTITY  "Content-ID:<andhardyharhar>"
DOCTYPE  "MEMO"       "Content-ID:<mahilagorilla>"
PUBLIC   "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN"
         "Content-ID:<mrpeebles>"
ENTITY   "%ISOnum"    "Content-ID=<wallygator>"
ENTITY   "MyEnding"   "Content-ID:<mrwhoopy>"
ENTITY   "Legal"      "Content-ID:<johnnyquest>"
SEMANTICS "large-print"  "DSSSL"  "Content-ID:<hodgyandbandit>"
```

```
--go-speed-racer
Content-Type:Application/SGML; charset=us-ascii
Content-ID:"<joejitsu>"

description of SGML declaration in Appendix A is included here

--go-speed-racer
Content-Type:Application/SGML; charset=us-ascii
Content-ID:"<mumbles>"

ISO 646 character set definition included here

--go-speed-racer
Content-Type:Application/SGML; charset=us-ascii
Content-ID:"<mastercylinder>"

description of Capacity in Appendix A is included here

--go-speed-racer
Content-Type:Application/SGML; charset=us-ascii
Content-ID:"<lippythelion>"

description of Syntax in Appendix A is included here

--go-speed-racer
Content-Type:Application/SGML; charset=us-ascii
Content-ID:"<underdog>"

Contents of ISO Registration Number 100//CHARSET ECMA-94 Right-hand Part
of Latin Alphabet Nr.1//ESC 2/13 4/1 included here

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <andhardyharhar>

include Document entity as described in Appendix C

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <mahilogorilla>

include DTD as described in Appendix B

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <mrpeebles>

ISO 8879-1986 Entity set included here

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <wallygator>

include entity set defined for %ISOnum

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <mrwhoopy>
```

```
include entity MyEnding as described in Appendix C

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <johnnyquest>

include entity Legal as described in Appendix C

--go-speed-racer
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <hodgyandbandit>

included here is a bunch of DSSSL-Lite

--go-speed-racer--
```

5.4 Sending a Catalog and a Single Non-Document Entity

This example describes what a server may send in response to a request for a non-document entity. All of the previous examples assume that the original request was for the document entity. SGML documents can get very deep and have a large number of external identifiers referenced in it. Likewise, the complete catalog for a document could also get very large (a "complete catalog" contains all of the external identifiers referenced in all of the document’s entities). There is no reason why the complete catalog has to be sent with the document entity. All that’s required are enough entries in the catalog for the client system to resolve references declared in the entity being transferred. For example, Appendix C.2 defines an entity called "Legal" which includes a reference to an entity called "MyEnding". A request for "Legal" would result in a Multipart/Related message that looks like this:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=let-go-of-my-leg;
              type="Application/SGML-Catalog"

--let-go-of-my-leg
Content-Type: Application/SGML-Catalog; charset=us-ascii

BASEURL  "http://www.bill.com/docs/memo/mine/dummy"
ENTITY   "Legal" "Content-ID:<bookemdano>"
ENTITY   "MyEnding"   "ending.sgml"

--let-go-of-my-leg
Content-Type: Application/SGML; charset=us-ascii
Content-ID: <bookemdano>

include entity "Legal" from Appendix C.2

--let-go-of-my-leg--
```

6. Security Considerations

SGML documents, like other compound documents, may contain entities whose media-types present security concerns, e.g. Application/PostScript. Further, SGML may contain explicit processing instructions for a presentation or composition system; use of such instructions present concerns similar to those of Application/PostScript.

The use of active media-types with Notation declarations can provide an opportunity for the sender to execute a script or other code on the recipient’s machine.

7. Acknowledgments

Thanks go to Andre Alguero, Steve DeRose, Chris Maden, and Bill Smith here at EBT for helping me with the content and structure of this document. Thanks also go out to Wayne Wohler of IBM for his help on SGML declarations, a most confusing topic.

8. References

[1] Wayne Wohler, "SGML declarations",
    http://www.sil.org/sgml/wlw11.html
[2] Eric van Herwijnen, "Practical SGML", Second Edition,
    Kluwer Academic Publishers, 1994, ISBN 0-7923-9434-8
[3] Charles F. Goldfarb, "The SGML Handbook",
    Oxford University Press, 1994, ISBN 0-19-853737-9
[4] R. Fielding, "Relative Uniform Resource Locators", RFC 1808
[5] E. Levinson, "The MIME Multipart/Related Content-Type", RFC ????
[6] Daniel W. Connolly, HTML 2.0 SGML declaration found at
    http://www.w3.org/hypertext/WWW/MarkUp/html-spec/html.decl
[7] T. Berners-Lee, "Universal Resource Identifiers in WWW:
    A Unifying Syntax for the Expression of Names and Addresses of
    Objects on the Network as used in the World-Wide Web", RFC 1630
[8] T. Berners-Lee, L. Masinter, and M. McCahill,
    "Uniform Resource Locators (URL)", RFC 1738
[10] Paul Grosso, "Entity Management", SGML Open Draft
     Technical Resolution 9401:1994
[11] Charles F. Goldfarb, "Entity Management in SGML", 11/30/93
[12] Sollins, K. and Masinter, L.,
     "Functional Requirements for Uniform Resource Names", RFC 1737
[13] N.Borenstein, "MIME (Multipurpose Internet Mail Extensions)
     Part One: Mechanisms for Specifying and Describing the Format of
     Internet Message Bodies", RFC 1521
[14] "ISO 8879:1986 Information processing - The and office systems -
     Standard Generalized Markup Language (SGML)",
     Geneva, 15 October 1986

9. Authors’ Address

Don Stinchfield
Electronic Book Technologies, Inc.
One Richmond Square
Providence, RI 02906
(401) 421-9550 x280
des@ebt.com

Appendix A: SGML declaration Used In The Examples

This Appendix contains the definitions for the SGML declaration, for the CAPACITY parameter, and for the SYNTAX parameter. The SGML declaration is a modified version of the one used for HTML 2.0 [6] - I changed the CAPACITY and SYNTAX declarations so that they referenced public identifiers. The following external identifiers are reference in the SGML declaration:

```
     o  BASESET "ISO 646:1983//
             CHARSET International Reference Version (IRV)//
             ESC 2/5 4/0"
     o  BASESET  "ISO Registration Number 100//
             CHARSET ECMA-94 Right-hand Part of Latin Alphabet Nr.1//
             ESC 2/13 4/1"
     o  CAPACITY   PUBLIC "-//EBT//CAPACITY CoolCaps 1.0//"
     o  SYNTAX  PUBLIC "-//EBT//SYNTAX  SinSyn 0.1//"
```

A.1 SGML declaration

```
<!SGML  "ISO 8879:1986"
--
      Copied this from the SGML declaration for HyperText Markup
Language (HTML).
--

CHARSET
         BASESET  "ISO 646-1983//CHARSET
                   International Reference Version
                   (IRV)//ESC 2/5 4/0"
         DESCSET  0   9   UNUSED
                  9   2   9
                  11  2   UNUSED
                  13  1   13
                  14  18  UNUSED
                  32  95  32
                  127 1   UNUSED
         BASESET  "ISO Registration Number 100//CHARSET
                  ECMA-94 Right-hand Part of
                  Latin Alphabet Nr.1//ESC 2/13 4/1"

         DESCSET  128  32   UNUSED
                  160  96    160

CAPACITY   PUBLIC "-//EBT//CAPACITY CoolCaps 1.0//"
SCOPE      DOCUMENT
SYNTAX     PUBLIC "-//EBT//SYNTAX  SinSyn 0.1//"

FEATURES
  MINIMIZE
    DATATAG  NO
    OMITTAG  YES
    RANK     NO
    SHORTTAG YES
```

```
  LINK
    SIMPLE   NO
    IMPLICIT NO
    EXPLICIT NO
  OTHER
    CONCUR   NO
    SUBDOC   NO
    FORMAL   YES
  APPINFO    "SDA"  -- conforming SGML document Access application
                    --
>
```

A.2 Capacity

```
TOTALCAP        150000
GRPCAP          150000
ENTCAP          150000
```

A.3 Syntax

```
SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
             17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
         BASESET  "ISO 646-1983//CHARSET
                   International Reference Version
                   (IRV)//ESC 2/5 4/0"
         DESCSET  0 128 0
         FUNCTION
                  RE          13
                  RS          10
                  SPACE       32
                  TAB SEPCHAR  9


         NAMING   LCNMSTRT ""
                  UCNMSTRT ""
                  LCNMCHAR ".-"
                  UCNMCHAR ".-"
                  NAMECASE GENERAL YES
                           ENTITY  NO
         DELIM    GENERAL  SGMLREF
                  SHORTREF SGMLREF
         NAMES    SGMLREF
         QUANTITY SGMLREF
                  ATTSPLEN 2100
                  LITLEN   1024
                  NAMELEN  72    -- somewhat arbitrary; taken from
                                internet line length conventions --
                  PILEN    1024
                  TAGLVL   100
                  TAGLEN   2100
                  GRPGTCNT 150
                  GRPCNT   64
```

Appendix B: DTD Used In The Examples

The DTD listed below is a modified version of the one found on page 33 of Eric vanHerwijnen's book called "Practical SGML" [2]. The following external identifier is used in the DTD:

```
     <!ENTITY % ISOnum PUBLIC
                "ISO 8879-1986//
                ENTITIES Numeric and Special Graphic//EN"
                "/usr/wcs/isonum.ent" >
```

The above definition is for a parameter entity and it contains both a public identifier and a system identifier. The examples have both in the catalog.

B.1 DTD

```
<!--  DTD for simple office memoranda                            -->
<!ENTITY % doctype "MEMO"  -- document type generic identifier -->
<!--        ELEMENTS    MIN     CONTENT      (EXCEPTIONS)        -->
<!ELEMENT   %doctype;   - -    ((TO & FROM),  BODY, CLOSE?)       >
<!ELEMENT   TO          - O    (#PCDATA)                          >
<!ELEMENT   FROM        - O    (#PCDATA)                          >
<!ELEMENT   BODY        - O    (P)*                               >
<!ELEMENT   P           - O    (#PCDATA | Q)*                     >
<!ELEMENT   PREF        - O    EMPTY                              >
<!ELEMENT   Q           - -    (#PCDATA)                          >
<!ELEMENT   CLOSE       - -    (#PCDATA)                          >
<!--        ELEMENTS    NAME   VALUE               DEFAULT       -->
<!ATTLIST   %doctype    STATUS (CONFIDEN|PUBLIC)   PUBLIC         >
<!ATTLIST   P           id     ID                  #IMPLIED       >
<!ATTLIST   PREF        refid  IDREF               #IMPLIED       >
<!ENTITY    %  ISOnum PUBLIC "ISO 8879-1986//ENTITIES Numeric and
                             Special Graphic//EN"
                             "/usr/wcs/isonum.ent">
%ISOnum;
```

Appendix C: SGML document Used In The Examples

The SGML document defined in this appendix is broken up into 3 parts: an SGML document entity and two SGML Entities. The SGML document entity contains references to external identifiers in the DOCTYPE and ENTITY declarations:

- This one contains both a public identifier and a system identifier:

```
          <!DOCTYPE MEMO  PUBLIC "-//EBT//DTD Released Memo//EN"
                                 "/usr/wcs/dtd/memo.dtd" [...] >
```

- This ENTITY declaration has system identifier and a system identifiers parameter:

```
          <!ENTITY  MyEnding SYSTEM "/usr/des/ending.sgm"  >
```

- This one specifies a system identifier without specifying a system identifier parameter (this is provided for in the SGML Standard for implementers that want to resolve System Identifiers from the entity name alone [3, p378]):

```
             <!ENTITY  Legal SYSTEM  >
```

C.1 SGML document entity

```
<!DOCTYPE MEMO  PUBLIC
     "-//EBT//DTD Released Memo//EN"
     "/usr/wcs/dtd/memo.dtd" [
<!ENTITY  Legal     SYSTEM  >
<!ENTITY  MyEnding  SYSTEM "/usr/des/ending.sgm"  >
] >
<MEMO>
<TO>Anaximander
<FROM>Cool Papa Shad
<BODY>
&Legal;
<P>Yo Anax, you've got a bizarre name!</P>
<P>&MyEnding;</P>
</BODY>
</CLOSE>
</MEMO>
```

C.2 Entity Named "Legal"

```
<P>If you or anyone you know tries to read this email then you're in
really big trouble!</P>
<P>You know this is the end of the document when you see &MyEnding;</P>
```

C.3 Entity Named "MyEnding"

```
<Q>Regards, Don</Q>
```