ENTITY MANAGEMENT

SGML Open Technical Resolution 9401:1997

(Amendment 2 to TR 9401)

Paul Grosso, ArborText, Chair, Entity Management Subcommittee, SGML Open

Revision date: 1997 September 10

Copyright © 1994, 1995, 1997 by SGML Open

Permission to reproduce parts or all of this information in any form is granted to SGML Open members provided that this information by itself is not sold for profit and that SGML Open is credited as the author of this information.

Two different but related issues pertaining to entity management impede interoperability of SGML documents:

A. that of interpreting external identifiers in entity declarations so that an SGML document can be processed by different vendors' tools on a single computer system, and

B. that of moving SGML documents to different computers in a way that preserves the association of external identifiers in entity declarations with the correct files or other storage objects.

While there are many important issues involved and a complete solution is beyond the current scope, the SGML Open membership agrees upon the enclosed set of conventions to address a useful subset of the complete problem. To address issue A, this resolution defines an entity catalog that maps an entity's external identifier and/or name to a file name, URL, or other storage object identifier. To address issue B, this resolution defines a simple interchange packaging scheme using an interchange catalog to associate a public identifier with each interchanged file.

Technical Resolution 9401:1994
Final Technical Resolution: 1994 August 9
Technical Resolution 9401:1995 (Amendment 1)
Final Technical Resolution: 1995 September 8
Technical Resolution 9401:1997 (Amendment 2)
Final Technical Resolution: 1997 September 10

Introduction

In order to use a variety of SGML tools in a variety of computer environments, there are two different but related problems to solve:

A. that of interpreting external identifiers in entity declarations so that an SGML document can be processed by different vendors' tools on a single computer system, and

B. that of moving SGML documents to different computers in a way that preserves the association of external identifiers in entity declarations with the correct files or other storage objects.

There are many important issues involved and a complete solution—possibly including work within the standards community—is beyond the current scope. However, the SGML Open membership agrees at this time upon a set of conventions that addresses a useful subset of the complete problem.

The short term solution for issue A defines an entity catalog that handles the simple cases of mapping an external entity's public identifier and/or entity name to a file name, URL, or other storage object identifier. This solution allows for a probably system-dependent (at least in the case of file names) but application-independent catalog. Though it does not handle all issues that a combination of a complete entity manager and storage manager addresses, it simplifies use of multiple products in a great majority of cases and can in some cases (e.g., with URLs) provide internet-wide, system-independent resolution of public identifiers.

While there are various interchange strategies already defined—including the SGML Document Interchange Format (SDIF) defined in ISO 9069—none are currently widely used or supported by enough readily accessible implementations. This resolution addresses issue B by defining a simple interchange packaging scheme using an interchange catalog to associate a public identifier with each interchanged file.

Issue A: a simple entity catalog format

To address the issue of multiple vendors' applications on a given system, this resolution defines a format for a probably system-dependent but application-independent entity catalog that maps external identifiers and/or entity names to file names. This catalog is used by an application's entity manager. This resolution does not dictate when an entity manager should access this catalog; for example, an application may attempt other mapping algorithms before or (if the catalog fails to produce a successful mapping) after accessing this catalog. The catalog has a standard format. Each application that uses it must provide the user with a mechanism for specifying how and when the catalog is to be accessed.

For the purposes of this resolution, the term **catalog** refers to the logical "mapping" information that may be physically contained in one or more catalog entry files. The catalog, therefore, is effectively an ordered list of (one or more) catalog entry files. It is up to the application to determine the ordered list of catalog entry files to be used as the logical catalog. (This resolution uses the term "catalog entry file" to refer to one component of a logical catalog even though a catalog entry file can be any kind of storage object or entity including—but not limited to—a table in a database, some object referenced by a URL, or some dynamically generated set of catalog entries.)

Each entry in the catalog associates a "Formal System Identifier" (FSI) with information about the external entity that appears in the SGML document. Formal System Identifiers (FSIs) are defined as part of the SGML General Facilities, currently part of the Technical Corrigendum to the HyTime standard ISO/IEC 10744. "Storage object identifiers" (such as file names) are a simple subset of all FSIs. ("Storage object identifier" is frequently abbreviated "s.o.i." below.) Valid FSIs include unpathed, relative, and absolute file names and URLs as well as FSIs with explicit storage managers (as defined in the SGML General Facilities). Most of the examples in this resolution will show s.o.i.s, but this resolution allows FSIs as the right hand side of most catalog entries. For example, the following are possible catalog entries that associate a public identifier with an s.o.i.:

```
PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN" "iso-lat1.gml"
PUBLIC "-//USA/AAP//DTD BK-1//EN" "aapbook.dtd"
PUBLIC "-//ACME//DTD Report//EN" "http://acme.com/dtds/report.dtd"
```

In addition to entries that associate public identifiers, a catalog entry can associate an entity name with an s.o.i. (or other FSI):

```
ENTITY "chips" "graphics\chips.tif"
```

Both types of entries can occur in a single catalog:

```
PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN" "iso-lat1.gml"
PUBLIC "-//ACME//DTD Report//EN" "http://acme.com/dtds/report.dtd"
ENTITY "graph1" "graphics\graph1.cgm"
```

The name field in an ENTITY type catalog entry gives the "entity name" as specified in the entity declaration of an entity whose entity text is specified by an external entity specification. [In an

external entity declaration, the “entity text” is the part that locates—via an external identifier—the entity’s replacement text—see clause 4.127 of the SGML standard. The term “replacement text” refers to the material that is to replace an entity reference—see clause 4.266—irrespective of the entity’s type (e.g., SGML, CDATA, NDATA).] Note that, if the entity name is a parameter entity name (as opposed to a general entity name), an initial percent sign (%), is part of the name. (The percent sign—which is the reference concrete syntax replacement for the “PERO” character—shall be used in the catalog regardless of the concrete syntax of the current document.) It should be noted that ENTITY type catalog entries will not match the reference to the external subset in a DOCTYPE or LINKTYPE declaration. The complete set of catalog entry types defined by this Resolution are: PUBLIC, ENTITY, NOTATION, SYSTEM, DOCTYPE, LINKTYPE, SGMLDECL, DTDDECL, DOCUMENT, DELEGATE, CATALOG, OVERRIDE, and BASE.

Furthermore, to provide for possible future extensions or other uses of this catalog, its format allows for “other information”—indicated by a “keyword” other than one of those defined by this Resolution—that is irrelevant to and ignored by this resolution.

The formal syntax for a catalog entry file is:

```
catalog = ps*, (catalog entry, (ps+, catalog entry)*, ps*)?

catalog entry = extended external identifier entry | no identifier entry |
                other information

other information = keyword, ps+, (symbol | non-symbol token | literal),
                            (ps+, (non-symbol token | literal))*

extended external identifier entry =
  (publicid keyword, ps+, public identifier, ps+, FSI specification) |
  (name keyword, ps+, entity name spec, ps+, FSI specification) |
  (noname keyword, ps+, FSI specification) |
  ("SYSTEM", ps+, system identifier, ps+, FSI specification) |
  ("DELEGATE", ps+, partial public identifier, ps+, FSI specification)

no identifier entry = "OVERRIDE", ps+, ("YES" | "NO")

partial public identifier = minimum literal

publicid keyword = "PUBLIC" | "DTDDECL"

name keyword = "ENTITY" | "DOCTYPE" | "LINKTYPE" | "NOTATION"

noname keyword = "SGMLDECL" | "DOCUMENT" | "BASE" | "CATALOG"

entity name spec = (symbol | non-symbol token | literal)

FSI specification = (symbol | non-symbol token | literal)

keyword = symbol
```

```
symbol = restricted system character+

non-symbol token = restricted system character*, special system character,
                   (restricted system character | special system character)*

literal =
  (LIT, system character+, LIT) |
  (LITA, system character+, LITA)

ps = s | comment

comment = COM, system character*, COM

special system character = "/" | "\" | "." | "<" | ">"

LIT = '"'   -- the double quote --

LITA = "'"   -- the single quote --

COM = "--"
```

where

1. **public identifier**, **system identifier**, **minimum literal**, and **s** are as defined in 8879 (and RS, RE, SPACE and SEPCHAR are as in the reference concrete syntax of 8879);
2. **system character** means (a) 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); (b) 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.
3. **restricted system character** means any character except the "null" character, the LIT character, the LITA character, those characters allowed in **s**, and any of the characters "\/.<>".

Additional requirements:

a. Recognition of the keywords must be case-insensitive.

b. Recognition of **keyword** and unquoted **argument**, **entity name spec**, and **FSI specification** tokens with respect to the COM delimiter shall be as defined in 8879. Briefly, the string "--" is recognized as the start of a comment if and only if this string constitutes the first two (or only) characters of a token and is always recognized as the end of a comment; however, see 8879 for the authoritative discussion.

c. Any **argument** other than the first that is part of **other information** and that would lexically be a valid keyword must be quoted. (This implies that, following an unrecognized keyword and its required initial [or only] argument, the first unquoted token that would be a lexically valid keyword shall in fact be interpreted as the next keyword.)

d. Limits on the length of any string of **system character**s must not preclude strings of any reasonable length; at a minimum, lengths up to 1024 must be supported.

e. This resolution does not formally place restrictions on the form of the FSIs in the catalog. It is the responsibility of the catalog creator and the end user to ensure compatibility among the catalog, the tools that will read the catalog, and the environment in which the catalog is used. In the interest of interoperability, this resolution does dictate that any **storage object identifier** that consists solely of alphanumeric characters, hyphen, period, and underscore must be treated as a file name (these are the characters in the POSIX portable file name character set).

f. If a **storage object identifier** specifies a relative path name, the path is relative to the location of the catalog entry file itself (unless a previous occurrence of a BASE entry has occurred in this catalog entry file, in which case the path specified by the s.o.i. is relative to the path given on the BASE entry).

This resolution only requires applications to handle storage object identifiers that specify file names. (Whether the s.o.i. can contain, for example, environment variables or special characters that are expected to be expanded further before resolving to a file name is not prescribed by this Resolution.) Applications may in addition recognize other types of storage object identifiers and Formal System Identifiers, as long as a storage object identifier that does not include characters other than letters, digits, hyphen, period, and underscore continues to be treated as a file name. Therefore, to avoid possible interpretation as something other than a file name, it is recommended (but not required) that file names be restricted to the characters just mentioned.

An entry in the catalog is interpreted as follows:

1. The PUBLIC keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text for an entity with the specified **public identifier**.

2. The ENTITY keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text for an entity with the entity name specified by the **entity name spec**.

3. The NOTATION keyword indicates that an entity manager should use the associated **storage object identifier** for a notation with the notation name specified by the **entity name spec**. This resolution does not address the form of the **storage object identifier** associated to a notation's external identifier or how an application makes use of it. Other resolutions or conventions outside the scope of this resolution may address such issues.

4. The SYSTEM keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text for an entity whose external identifier's system identifier is explicitly specified by the **system identifier**.

5. The DOCTYPE keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text (to be used as the external subset) for a doctype declaration whose document type name is specified by the **entity name spec**. Note

that a document type declaration that omits the optional external identifier (that points to the external subset) indicates the absence of an external subset; in this case, there is no entity reference to resolve, and no catalog lookup is performed.

6. The LINKTYPE keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text (to be used as the external subset) for a linktype declaration whose link type name is specified by the **entity name spec**. Note that a link type declaration that omits the optional external identifier (that points to the external subset) indicates the absence of an external subset; in this case, there is no entity reference to resolve, and no catalog lookup is performed.

7. The SGMLDECL keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text to be used as the SGML declaration.

8. The DTDDECL keyword indicates that an entity manager should use the associated **storage object identifier** to locate the replacement text to be used as the SGML declaration. Note that the **public identifier** in a DTDDECL entry is meant to match a public identifier given as part of the doctype declaration to reference the external subset.

9. The DOCUMENT keyword indicates that an entity manager should use the associated **storage object identifier** to locate the entity in which parsing begins.

10. The DELEGATE keyword indicates that external identifiers with a public identifier that has **partial public identifier** as a prefix should be resolved using a catalog is specified by the associated **storage object identifier**.

11. The CATALOG keyword indicates that an entity manager should use the associated **storage object identifier** to locate an additional catalog entry file to be processed after the current catalog entry file.

12. The OVERRIDE keyword specifies whether to use the “prefer system id” mode or not for the search strategy (see below for more discussion).

13. The BASE keyword specifies that relative storage object identifiers in the right hand side of entries following this entry in the current catalog entry file should be resolved relative to the **storage object identifier** of this BASE entry.

The declaration of every external entity includes an entity name. (For the purposes of this discussion and the table below, we consider the term “entity name” to encompass also the doctype name from the document type declaration and the link type name from the link type declaration.) It may, in addition, associate a public identifier and/or a system identifier with the external entity.

When doing a catalog lookup, an entity manager generally uses whatever is available from among the entity declaration’s system identifier, public identifier, and entity name to find catalog entries that match the given information. A match in one catalog entry file will take precedence over any match in a later catalog entry file (and, in fact, the entity manager need not process subsequent catalog entry files once a match has occurred). A more specific matching entry in one catalog

entry file will take priority over a less specific matching entry in the same catalog entry file. For this purpose, the order of specificity of match (most specific first) is:

1. SYSTEM type entries;
2. PUBLIC type entries;
3. DELEGATE entries ordered by the length of the prefix, longest first;
4. ENTITY, DOCTYPE, LINKTYPE, and NOTATION type entries.

Within any given category of equal specificity, matches maintain the order of their entries in the catalog entry file so that the first such match will take priority.

Generally, when a system identifier is specified in an external entity declaration, it can be trusted to be a valid s.o.i. However, in some circumstances (such as when the document was generated on another system, when the document was generated in another location on the same system, or when some files referenced by system identifiers have had their locations changed since the document was generated), the specified system identifiers may not be valid. For this or other reasons, preferring the public identifier or entity name over the system identifier may be the preferred way of accessing the entity. Therefore, this resolution defines two modes for using the above search strategy when an external identifier has an explicit system identifier. (Furthermore, a SYSTEM catalog entry can be used to map an explicit system identifier given in an external entity declaration into any s.o.i; a matching SYSTEM type entry would take precedence over a PUBLIC type entry regardless of the search mode strategy.) The two search modes are:

1. If system identifiers are preferred and there is no matching SYSTEM type entry, then the system identifier is used as the s.o.i. regardless of the entity name and any public identifier. This resolution does not specify what happens if a preferred system identifier does not identify an accessible storage object; an application may look up the public identifier and/or entity name to find another s.o.i., or it may simply report an error. An application should at least have the option of issuing a warning if the system identifier fails in this mode.
2. If public identifiers and entity names are preferred and there is no matching SYSTEM type entry, the system identifier is used as the s.o.i. only if no mapping can be found in the catalog entry file for either the public identifier (if a public identifier was specified) or for the entity name.

An application must provide some way (e.g., a runtime argument, environment variable, preference switch) that allows the user to specify which of these modes to use in the absence of any occurrences of an OVERRIDE catalog entry.

The OVERRIDE catalog entry type can be used within any catalog entry file to indicate for any set of catalog entries whether they should be able to be used in matches that may override an explicit system identifier. Each occurrence of an OVERRIDE entry specifies the search strategy mode for subsequent entries up to the next OVERRIDE entry or the end of the current catalog entry file. A PUBLIC, DELEGATE, ENTITY, DOCTYPE, LINKTYPE or NOTATION entry encountered when OVERRIDE is “YES” (corresponding to the mode where public identifiers and entity names are preferred) will be considered for possible matching whether or not the external identifier has

an explicit system identifier. A PUBLIC, DELEGATE, ENTITY, DOCTYPE, LINKTYPE or NOTATION entry encountered when OVERRIDE is "NO" (corresponding to the mode where system identifiers are preferred) will be ignored during lookups for which the external identifier has an explicit system identifier. No other entry types are affected by the OVERRIDE catalog entry. The initial search strategy in force at the beginning of each catalog entry file depends on the preference as determined by the application (possibly under user control).

When attempting matches for DELEGATE type catalog entries, the entity's public identifier is compared to the **partial public identifier** of the DELEGATE catalog entry looking for partial public identifiers that are initial substring matches of the entity's public identifier. If this catalog entry file produces any such matches, the right hand side of all such matching entries are used, in order from longest **partial public identifier** match to shortest, to generate a new complete logical catalog (i.e., a newly specified list of catalog entry files) that replaces the current catalog.

The catalog lookup process for this entity continues with this new (replacement) catalog, ignoring for the purposes of this entity any other entries in the current catalog entry file as well as any subsequent catalog entry files that may have been part of the previous list of catalog entry files. This newly defined catalog is then processed in much the same manner as if it had been the originally specified catalog; however, only the entity's public identifier is considered as the information available for lookup—its entity name and system identifier (if any) are not available during lookup in any "delegated to" catalog. Lookup for subsequent public identifiers is unaffected by this process; that is, the effect of this replacement catalog holds only for the lookup of the current entity's public identifier.

The CATALOG entry can be used to insert new catalog entry files into the current list of catalog entry files. The **storage object identifier** on a CATALOG entry is used to locate another catalog entry file that is processed after the current catalog entry file if the current catalog entry file does not provide a match. Multiple CATALOG entries are allowed, and the referenced catalog entry files will be inserted into the current catalog list in order. Note that the effect of any CATALOG entry would occur only after all other entries in this catalog entry file have been considered.

1 The use of hyphens or colons in the ISO owner identifier

Since this resolution pertains to public identifiers, it addresses one additional detail about public identifiers. ISO 8879 is inconsistent about the use of hyphens and colons in **ISO owner identifier**s. Clause 10.2.1.1 of 8879:1986 (unamended) has a note indicating that the ISO owner identifier for the SGML standard is "ISO 8879–1986". Production [171] of clause 13 indicates that the minimum literal in the SGML declaration must be "ISO 8879–1986". While Amendment 1 of 8879 does not alter clause 10.2.1.1, it does alter production [171] of clause 13 to say that the minimum literal in the SGML declaration should be "ISO 8879:1986". This has lead to the propagation of both the dash and the colon in ISO owner identifiers. In the interests of interoperability, this SGML Open resolution requires that all products accept either form as a valid ISO owner identifier. Note, however, that this should not be construed to mean that a public identifier using one form should necessarily cause a catalog lookup match to succeed with a public identifier using the other form; while this resolution requires SGML systems to accept

either form as valid, in practice, two entries (differing only by the single ":" or "–" character) may be needed in the catalog if both forms should refer to the same storage object identifier.

2 Referencing the implied SGML declaration

The SGML standard allows for an SGML declaration to be included explicitly in a document or to be implied by the processing system. This Resolution defines two ways to specify the implied SGML declaration: the SGMLDECL catalog entry type and the DTDDECL catalog entry type. Note that, in the DTDDECL method, the implied SGML declaration depends on information in the remainder of the document. Since the SGML declaration must be processed before a parser can interpret the prolog and document instance set, an implementation may choose to determine the SGML declaration with a preprocessor that scans the document for the relevant information. In any case, once it has been determined whether an explicit SGML declaration is present and, if not, how to locate the implied SGML declaration, parsing begins at the start of the document.

In many situations, the appropriate SGML declaration can be inferred from the "DTD" in use. This is especially common in the case that the external subset referenced in the doctype declaration is a publicly distributed entity. Therefore, this Resolution adds the capability to associate an SGML declaration with a "DTD" referenced by a PUBLIC identifier. In particular, if there is no explicit SGML declaration and the doctype declaration uses a PUBLIC identifier to reference the external subset (commonly known as "the DTD"), then the catalog will be searched for a DTDDECL entry whose **public identifier** field matches the public identifier of the external subset, and the associated s.o.i. will be used to locate the default SGML declaration to be used.

If there is no explicit SGML declaration and no DTDDECL entry was applicable, then the catalog will be searched for the first SGMLDECL entry, and its s.o.i. will be used to locate the default SGML declaration to be used. The use of an SGMLDECL catalog entry, in fact, is the preferred method of indicating the SGML declaration when an SGML declaration is part of a transfer package but is not transmitted as the initial part of the document entity itself.

Issue B: an interchange packaging scheme

The issue of interchanging a set of files among different systems can be partially addressed by an interchange packaging scheme that includes an interchange catalog that associates external identifiers with the various files in the interchange package. This resolution, which assumes the catalog format defined above, describes such a scheme.

This resolution does not support the use of explicitly specified system identifiers; that is, an external entity's declaration may specify a public identifier or it may use the SYSTEM keyword with no system identifier (in which case the entity's name will be used to do a catalog lookup for a matching catalog entry indicated by the ENTITY keyword). This resolution assumes a transmission medium that allows for the interchange of names for the various files in the interchange package.

The actual transmission medium and details of writing and reading the interchange package are irrelevant. This resolution assumes that there exists a single location (e.g., directory) on the receiving system that already contains the set of interchanged files. (The generation of such an interchange package by the sending system is not explicitly discussed, but it is assumed that this discussion about receiving and interpreting an interchange package will make clear what is necessary to do on the sending system.) In this resolution, the phrase "interchange package" refers to this set of files in this location and "interchange directory" refers to this location.

An interchange package must have at least one file that shall function as the interchange package's catalog. This catalog entry file must have a mapping for all files in the interchange package. That is, for each file in the interchange package (other than this catalog file), there must be a catalog entry whose s.o.i. identifies the file.

To determine what file in the interchange package shall be used as the catalog, an application shall use the following algorithm (or functional equivalent):

1. If the document entity's s.o.i. is somehow known to the application, the application should first look for a storage object whose s.o.i. is ***docname*.soc** where "docname" is the "base name" of the document entity's s.o.i. An s.o.i.'s base name is determined as follows:

   a. within the s.o.i., locate the last (rightmost) character that is either "/" or "\" if any;

   b. within the string to the right of this character (or within the entire s.o.i. if there are no occurrences of either the "/" or "\" character), locate the last (rightmost) "." character (called the dot, period, or full stop character) if any;

   c. the string consisting of all characters in the s.o.i. up to but not including this "." character (or the entire s.o.i. if the previous step found no "." character) shall be the s.o.i.'s base name.

   (The base name determination algorithm is optimized for URLs and certain common file naming schemes; however, on all operating systems, this algorithm may fail to be useful unless appropriate naming conventions are followed.)

If the ***docname*.soc** s.o.i. names a relative (as opposed to absolute) location, it shall be resolved into an absolute location using the same process used to resolve the document entity's relative s.o.i. into an absolute one. (This resolution does not specify how the application may know the document entity's file name prior to reading the catalog. It may be given to the application via a command line option or a via a user dialog. Note, of course, that the DOCUMENT entry in the catalog cannot be used to determine the document entity's file name for the purposes of determining the catalog's file name.)

2. Then, look for a file whose name is **catalog**.

3. Finally, look for a file whose name is **catalog.soc**.

In the second step above, if the letter case of file names is significant for the operating system involved, then first the name **catalog** in all lower case and then the name **CATALOG** in all upper case will be tried (and no mixed case combinations are tried). Throughout the entire algorithm, as soon as a readable file is found, that file is used and no further names are tried.

Ordinarily, the catalog should include a single entry of the DOCUMENT type whose s.o.i. identifies the file in the interchange package that is the document entity in which parsing begins, if any such entity exists in this interchange package. (Some interchange packages may not include such an entity, for example, if the interchanged files are a set of entity declaration files.) Although it does not prohibit such interchange, this resolution does not make explicit allowance for including multiple documents in a single interchange. To ensure maximum portability, each interchange package should consist of at most one document. (Since this resolution does not address details of actual transmissions, it does not prohibit multiple interchange packages within a single transmission.)

Provided that the interchange package's catalog has an unambiguous entry for each file named in the interchange package, an interchange package is valid even if the receiver must modify the s.o.i.s in his/her copy of the catalog so that they are valid on the receiving system. However, when the sending and receiving systems have compatible naming schemes, files in the destination location may be given the same names as they had on the sending system. This possibility is more likely because relative paths in s.o.i.s are relative to the catalog file and therefore relative to the top level of the interchange directory. If the receiving system is unknown or incompatible with the sending system, the sender may wish to construct an interchange package with names that are most likely to be valid on the widest variety of systems. (For example, an interchange package with file names of no more than eight alphanumeric characters—and therefore no directory hierarchy—should be maximally portable. However, this resolution does not impose any such restrictions since, in practice, it will often be known what the receiving system can handle, and it will be preferable to take advantage of its capabilities.)