Applications Area Working Group
Internet-Draft
Obsoletes: 1738 (if approved)
Intended status: Standards Track
Expires: May 5, 2016

M. Kerwin
QUT

November 2, 2015

The file URI Scheme
draft-ietf-appsawg-file-scheme-04

Abstract

This document specifies the "file" Uniform Resource Identifier (URI) scheme, obsoleting the definition in RFC 1738.

It attempts to define a common core which is intended to interoperate across the broad spectrum of existing implementations, while at the same time documenting other current practices.

Note to Readers (To be removed by the RFC Editor)

This draft should be discussed on the IETF Applications Area Working Group discussion list <apps-discuss@ietf.org>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on May 5, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

A file URI identifies a file on a particular file system. It can be used in discussions about the file, and if other conditions are met it can be dereferenced to directly access the file.

The file URI scheme is not coupled with a specific protocol, nor with a specific media type. See Section 3 for a discussion of operations that can be performed on a file URI.

This document defines a syntax that is compatible with most extant implementations, while attempting to push towards a stricter subset of "ideal" constructs. In many cases it simultaneously acknowledges and deprecates some less common or outdated constructs.

1.1. History

The file URI scheme was first defined in [RFC1630], which, being an informational RFC, does not specify an Internet standard. The definition was standardised in [RFC1738], and the scheme was registered with the Internet Assigned Numbers Authority (IANA); however that definition omitted certain language included by former that clarified aspects such as:

- the use of slashes to denote boundaries between directory levels of a hierarchical file system; and

- the requirement that client software convert the file URI into a file name in the local file name conventions.

The Internet draft [I-D.hoffman-file-uri] was written in an effort to keep the file URI scheme on standards track when [RFC1738] was made obsolete, but that draft expired in 2005. It enumerated concerns arising from the various, often conflicting implementations of the scheme. It serves as the spiritual predecessor of this document.

Additionally the WHATWG defines a living URL standard [WHATWG-URL], which includes algorithms for interpreting file URIs (as URLs).

1.2. Similar Technologies

The Universal Naming Convention (UNC) [MS-DTYP] defines a string format that can perform a similar role to the file URI scheme in describing the location of files. A UNC filespace selector string has three parts: host, share, and path; see Appendix E. This document describes but does not specify a means of translating between UNC filespace selector strings and file URIs in Appendix C.2.

1.3. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Throughout this document the term "local" is used to describe files that can be accessed directly through the local file system. It is important to note that a local file may not be physically located on the local machine, for example if a networked file system is transparently mounted into the local file system.

2. Syntax

The file URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], including the core ABNF syntax rule "ALPHA" defined by that specification, and importing the "userinfo", "host", "authority" and "path-absolute" rules from [RFC3986] (as updated by [RFC6874].)

Please note Appendix C that lists some other commonly seen but nonstandard variations.

```
   file-URI       = file-scheme ":" file-hier-part

   file-scheme    = "file"

   file-hier-part = "//" auth-path
                  / local-path

   auth-path      = [ file-auth ] path-absolute

   local-path     = path-absolute

   file-auth      = [ userinfo "@" ] host
```

The syntax definition above is different from those given in [RFC1630] and [RFC1738] as it is derived from the generic syntax of [RFC3986], which post-dates all previous specifications.

As a special case, the "auth-path" rule can match the string "localhost" or the empty string in the URI's authority component; this is interpreted as "the machine from which the URI is being interpreted," exactly as if no authority was present.

Systems exhibit different levels of case-sensitivity. Unless the file system is known to be case-insensitive, implementations MUST maintain the case of file and directory names when translating file

URIs to and from the local system's representation of file paths, and any systems or devices that transport file URIs MUST NOT alter the case of file URIs they transport.

3. Operations on file URIs

Implementations SHOULD, at a minimum, provide a read-like operation to return the contents of a file located by a file URI. Additional operations MAY be provided, such as writing to, creating, and deleting files. See the POSIX file and directory operations [POSIX] for examples of standardized operations that can be performed on files.

File URIs can also be translated to and from other, similar constructs, such as local file paths or UNC strings.

A file URI can only be dereferenced or translated to a local file path if it is local. A file URI is considered "local" if it has a blank or no authority, or the authority is the special string "localhost".

This specification neither defines nor forbids a mechanism for accessing non-local files. See SMB [MS-SMB], NFS [RFC7530], NCP [NOVELL] for examples of protocols that can be used to access files over a network. Also see Appendix C.2 for a discussion on translating non-local file URIs to and from UNC stings.

3.1. Translating Local File Path to file URI

Below is an algorithmic description of the process used to convert a file path to a URI; see Section 4.

1. Resolve the file path to its fully qualified absolute form.

2. Initialise the URI with the "file:" scheme identifier.

3. If including an empty authority field, append the "//" sigil to the URI.

4. Append a slash character "/" to the URI, to signify the path root.

5. For each directory in the path after the root:

    1. Transform the directory name to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].

2. Append the transformed segment and a delimiting slash character "/" to the URI.

6. If the path includes a file name:

   1. Transform the file name to a path segment as above.

   2. Append the transformed segment to the URI.

Differences from RFC 1738

In [RFC1738] a file URL always started with the token "file://", followed by an (optionally blank) authority and a "/". That "/" was not considered part of the path. This implies that the correct encoding for a file path in a UNIX-like environment would have been:

```
  token     + authority + slash + path
= "file://" + ""        + "/"   + "/path/to/file.txt"
= "file:////path/to/file.txt"
```

However that construct was never observed in practice, and in fact would have collided with the eventual encoding of UNC strings in URIs described in Appendix C.3.

3.2. Translating Non-local File Path to file URI

Translating a non-local file path, including a UNC string, to a file URI follows the same basic algorithm as for local files, above, except that the authority MUST refer to the network-accesible node that hosts the file.

For example, in a clustered OpenVMS Files-11 system the authority would contain the node name. Where the original node reference includes a username and password in an access control string, they MAY be transcribed into the userinfo field of the authority ([RFC3986], Section 3.2.1), security considerations (Section 6) notwithstanding.

See Appendix C.2 for an explicit handling of UNC strings.

3.3. Incompatible File Paths

Some conventional file path formats are known to be incompatible with the file URI scheme.

3.3.1. Win32 Namespaces

The Microsoft Windows API defines Win32 Namespaces [Win32-Namespaces] for interacting with files and devices using Windows API functions. These namespaced paths are prefixed by "\\?\" for Win32 File Namespaces and "\\.\" for Win32 Device Namespaces. There is also a special case for UNC file paths in Win32 File Namespaces, referred to as "Long UNC", using the prefix "\\?\UNC\".

This specification does not define a mechanism for translating namespaced paths to or from file URIs.

4. Encoding

To avoid ambiguity, a file URI SHOULD be transported as an Internationalized Resource Identifier (IRI) [RFC3987], or as a URI with non-ASCII characters encoded according to the UTF-8 character encoding [STD63] and percent-encoded as needed ([RFC3986], Section 2.5).

The encoding of a file URI depends on the file system that stores the identified file. If the file system uses a known non-Unicode character encoding, the path SHOULD be converted to a sequence of characters from the Universal Character Set [ISO10646] normalized according to Normalization Form C (NFC) [UTR15], before being translated to a file URI, and conversely a file URI SHOULD be converted back to the file system's native encoding when dereferencing or translating to a file path.

> Note that many modern file systems encode directory and file names as arbitrary sequences of octets. In those cases, the representation as an encoded string often depends on the user's localization settings, or defaults to UTF-8 [STD63].

When the file system's encoding is not known the file URI SHOULD be transported as an Internationalized Resource Identifier (IRI) [RFC3987] to avoid ambiguity. See Appendix D for examples.

5. Origins

As per [RFC6454], Section 4, when determining the origin of a file URI implementations MAY return an implementation-defined value.

Historically, user agents have granted content from the file URI scheme a tremendous amount of privilege. However, granting all local files such wide privileges can lead to privilege escalation attacks. Some user agents have had success granting local files directory-based privileges, but this approach has not been widely adopted.

Other user agents use globally unique identifiers for each file URI, which is the most secure option.

6. Security Considerations

There are many security considerations for URI schemes discussed in [RFC3986].

File access and the granting of privileges for specific operations are complex topics, and the use of file URIs can complicate the security model in effect for file privileges. Software using file URIs MUST NOT grant greater access than would be available for other file access methods.

File systems typically assign an operational meaning to special characters, such as the "/", "\", ":", "[", and "]" characters, and to special device names like ".", "..", "...", "aux", "lpt", etc. In some cases, merely testing for the existence of such a name will cause the operating system to pause or invoke unrelated system calls, leading to significant security concerns regarding denial of service and unintended data transfer. It would be impossible for this specification to list all such significant characters and device names. Implementers MUST research the reserved names and characters for the types of storage device that may be attached to their application and restrict the use of data obtained from URI components accordingly.

Additionally, as discussed in the HP OpenVMS Systems Documentation <http://h71000.www7.hp.com/doc/84final/ba554_90015/ch03s09.html> "access control strings include sufficient information to allow someone to break in to the remote account, [therefore] they create serious security exposure." In a similar vein, the presence of a password in a "user:password" userinfo field is deprecated by [RFC3986]. As such, the userinfo field of a file URI, if present, MUST NOT contain a password.

7. IANA Considerations

This document defines the following URI scheme, so the "Permanent URI Schemes" registry has been updated accordingly. This registration complies with [BCP35].

Scheme name:
file

Status:
permanent

Applications/protocols that use this scheme name:
Commonly used in hypertext documents to refer to files without depending on network access. Supported by major browsers.

Windows API (PathCreateFromUrl, UrlCreateFromPath).

Perl LWP.

Contact:
Matthew Kerwin <matthew.kerwin@qut.edu.au>

Change Controller:
This scheme is registered under the IETF tree. As such, the IETF maintains change control.

[RFC Editor Note: Replace XXXX with this RFC's reference.]

8. Acknowledgements

This specification is derived from [RFC1738], [RFC3986], and [I-D.hoffman-file-uri] (expired); the acknowledgements in those documents still apply.

Additional thanks to Dave Risney, author of the informative IE Blog article <http://blogs.msdn.com/b/ie/archive/2006/12/06/file-uris-in-windows.aspx>, and Dave Thaler for their comments and suggestions.

9. References

9.1. Normative References

[BCP35] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <http://www.rfc-editor.org/info/bcp35>.

[ISO10646] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO/IEC 10646:2003, December 2003.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <http://www.rfc-editor.org/info/rfc3987>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874, February 2013, <http://www.rfc-editor.org/info/rfc6874>.

[UTR15] Davis, M. and K. Whistler, "Unicode Normalization Forms", August 2012, <http://unicode.org/reports/tr15/tr15-18.html>.

9.2. Informative References

[Bug107540] Bugzilla@Mozilla, "Bug 107540", October 2007, <https://bugzilla.mozilla.org/show_bug.cgi?id=107540>.

[I-D.hoffman-file-uri] Hoffman, P., "The file URI Scheme", draft-hoffman-file-uri-03 (work in progress), January 2005.

[MS-DTYP] Microsoft Open Specifications, "Windows Data Types, 2.2.56 UNC", January 2013, <http://msdn.microsoft.com/en-us/library/gg465305.aspx>.

[MS-NBTE] Microsoft Open Specifications, "NetBIOS over TCP (NBT) Extensions", May 2014, <http://msdn.microsoft.com/en-us/library/dd891412.aspx>.

[MS-SMB] Microsoft Open Specifications, "Server Message Block (SMB) Protocol", January 2013, <http://msdn.microsoft.com/en-us/library/cc246231.aspx>.

[NOVELL] Novell, "NetWare Core Protocols", 2013, <http://www.novell.com/developer/ndk/netware_core_protocols.html>.

[POSIX] IEEE, "IEEE Std 1003.1, 2013 Edition", 2013.

[RFC1630] Berners-Lee, T., "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, DOI 10.17487/RFC1630, June 1994, <http://www.rfc-editor.org/info/rfc1630>.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, DOI 10.17487/RFC1738, December 1994, <http://www.rfc-editor.org/info/rfc1738>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/std63>.

[WHATWG-URL] WHATWG, "URL Living Standard", May 2013, <http://url.spec.whatwg.org/>.

[Win32-Namespaces] Microsoft Developer Network, "Naming Files, Paths, and Namespaces", June 2013.

Appendix A. Example URIs

The syntax in Section 2 is intended to support file URIs that take the following forms:

Local files:

- "file:///path/to/file"

  A traditional file URI for a local file, with an empty authority. This is the most common format in use today.

- "file:/path/to/file"

  The minimal representation of a local file, with no authority field and an absolute path that begins with a slash "/".

Non-local files:

- "file://host.example.com/path/to/file"

  The representation of a non-local file, with an explicit authority.

Appendix B. System-specific Operations

This appendix is not normative; it highlights some observed behaviours and provides system-specific guidance for interacting with file URIs and paths.

B.1. POSIX Systems

There is little to say about POSIX file systems; the file URI structure already closely resembles POSIX file paths.

B.2. DOS- and Windows-Like Systems

When mapping a DOS- or Windows-like file path to a file URI, implementations typically map the drive letter (e.g. "c:") into the first path segment.

See Appendix C.1 for explicit (but non-normative and strictly optional) rules for interacting with DOS- or Windows-like file paths and URIs.

B.3. Mac OS X Systems

The HFS+ file system uses a non-standard normalization form, similar to Normalization Form D. Take care when transforming HFS+ file paths to and from URIs using Normalization Form C Section 4.

B.4. OpenVMS Files-11 Systems

When mapping a VMS file path to a file URI, map the device name into the first path segment. Note that the dollars sign "$" is a reserved character per the definition in [RFC3986], Section 2.2, so should be percent-encoded if present in the device name.

If the VMS file path includes a node reference, use that as the authority. Where the original node reference includes a username and password in an access control string, they can be transcribed into the userinfo field of the authority ([RFC3986], Section 3.2.1), security considerations (Section 6) notwithstanding.

Appendix C. Nonstandard Syntax Variations

These variations may be encountered for historical reasons, but are not supported by the normative syntax of Section 2.

This appendix is not normative.

C.1. DOS and Windows Drive Letters

On Windows- or DOS-based file systems a absolute file path can begin with a drive letter. To facilitate this, the "local-path" rule in Section 2 can be replaced with the following:

```
local-path     = [ drive-letter ] path-absolute

drive-letter   = ALPHA ":"
```

This is intended to support URIs of the form:

o "file:c:/path/to/file"

  The minimal representation of a local file in a DOS- or Windows-based environment, with no authority field and an absolute path that begins with a drive letter.

URIs of the form "file:///c:/path/to/file" are already supported by the "path-absolute" rule.

Note that comparison of drive letters in DOS or Windows file paths is case-insensitive. Some implementations therefore canonicalize drive letters in file URIs by converting them to uppercase.

C.1.1. Relative Paths

In DOS- or Windows-based file systems, relative paths beginning with a slash "/" should be resolved relative to the drive letter, and resolution of ".." dot segments (per Section 5.2.4 of [RFC3986]) should not ever overwrite the drive letter.

e.g.:

```
base:       file:///c:/path/to/file.txt
rel. URI:   /some/other/thing.bmp
resolved:   file:///c:/some/other/thing.bmp

base:       file:///c:/foo.txt
rel. URI:   ../../bar.txt
resolved:   file:///c:/bar.txt
```

Relative paths with a drive letter followed by a character other than a slash (e.g. "c:bar/baz.txt" or "c:../foo.txt") should not be accepted as dereferenceable URIs in DOS or Windows systems.

C.1.2. Vertical Bar Character

Historically some implementations have used a vertical line character "|" instead of a colon ":" in the drive letter construct. [RFC3986] forbids the use of the vertical line, however it may be necessary to interpret or update old URIs.

For interpreting such URIs, the "auth-path" and "local-path" rules in Section 2 and the "drive-letter" rule above are replaced with the following:

```
auth-path      = [ file-auth ] path-absolute
               / [ file-auth ] file-absolute

local-path     = [ drive-letter ] path-absolute
               / file-absolute

file-absolute  = "/" drive-letter path-absolute

drive-letter   = ALPHA ":"
               / ALPHA "|"
```

This is intended to support URIs of the form:

- "file:///c|/path/to/file"
- "file:/c|/path/to/file"
- "file:c|/path/to/file"

  Regular DOS or Windows file URIs, with vertical line characters in the drive letter construct.

To update such an old URI, replace the vertical line "|" with a colon ":".

C.2. UNC Strings

A UNC filespace selector string can be directly translated to a URI; see Section 4. The following is an algorithmic description of the process of translating a UNC string to a file URI:

1. Initialise the URI with the "file:" scheme identifier.
2. Append the authority:
   1. Append the "//" authority sigil to the URI.
   2. Append the hostname field of the UNC string to the URI.
3. Append the sharename:
   1. Transform the sharename to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].
   2. Append a delimiting slash character "/" and the transformed segment to the URI.
4. For each objectname:
   1. Transform the objectname to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].
   2. Append a delimiting slash character "/" and the transformed segment to the URI.

Example:

```
UNC String:   \\host.example.com\Share\path\to\file.txt
URI:          file://host.example.com/Share/path/to/file.txt
```

C.3. UNC Paths

It is common to encounter file URIs that encode entire UNC strings in the path, usually with all backslash "\" characters replaced with slashes "/".

To interpret such URIs, the "auth-path" rule in Section 2 is replaced with the following:

```
auth-path      = [ file-auth ] path-absolute
               / unc-authority path-absolute

unc-authority  = 2*3"/" [ userinfo "@" ] file-host

file-host      = inline-IP / IPv4address / reg-name

inline-IP      = "%5B" ( IPv6address / IPvFuture ) "%5D"
```

This syntax uses the "userinfo", "IPv4address, "IPv6address", "IPvFuture", and "reg-name` rules from [RFC3986].

Note that the "file-host" rule is the same as "host" but with percent-encoding applied to "[" and "]" characters.

This extended syntax is intended to support URIs that take the following forms, in addition to those in Appendix A:

Non-local files:

o "file:////host.example.com/path/to/file"

The "traditional" representation of a non-local file, with an empty authority and a complete (transformed) UNC string in the path.

o "file://///host.example.com/path/to/file"

As above, with an extra slash between the empty authority and the transformed UNC string, conformant with the definition from [RFC1738]. This representation is notably used by the Firefox web browser. See Bugzilla#107540 [Bug107540].

It also further limits the set of file URIs that can be translated to a local file path to those with a path that does not encode a UNC string.

C.4. Backslash as Separator

Historically some implementations have copied entire file paths into the path components of file URIs. Where DOS or Windows file paths were copied thus, resulting URI strings contained unencoded backslash "\" characters, which are forbidden by both [RFC1738] and [RFC3986].

It may be possible to translate or update such an invalid file URI by replacing all backslashes "\" with slashes "/", if it can be determined with reasonable certainty that the backslashes are intended as path separators.

Appendix D. Example of IRI vs Percent-Encoded URI

The following examples demonstrate the advantage of encoding file URIs as IRIs to avoid ambiguity (see Section 4).

Example: file IRI:

```
| Bytes of file IRI in a UTF-8 document:
|    66 69 6c 65 3a 43 3a 2f 72 65 c3 a7 75 2e 74 78 74
|    f  i  l  e  :  c  :  /  r  e  ( c ) u  .  t  x  t
|
| Interpretation:
|    A file named "recu.txt" with a cedilla on the "c", in the
|    directory "C:\" of a DOS or Windows file system.
|
| Character value sequences of file paths, for various file system
| encodings:
|
|  o UTF-16 (e.g. NTFS):
|       0043 003a 005c 0072 0065 00e7 0075 002e 0074 0078 0074
|
|  o Codepage 437 (e.g. MS-DOS):
|       43   3a   5c   72   65   87   75   2e   74   78   74
```

Counter-example: ambiguous file URI:

```
| Percent-encoded file URI, in any ASCII-compatible document:
|    "file:///%E3%81%A1"
|
| Possible interpretations of the file name, depending on the
| encoding of the file system:
|
|  o UTF-8:
|       <HIRAGANA LETTER TI (U+3061)>
|
|  o Codepage 437:
|       <GREEK SMALL LETTER PI (U+03C0)> +
|       <LATIN SMALL LETTER U WITH DIAERESIS (U+00FC)> +
|       <LATIN SMALL LETTER I WITH ACUTE (U+00ED)>
|
|  o EBCDIC:
|       "Ta~"
|
| etc.
```

Appendix E. UNC Syntax

The UNC filespace selector string is a null-terminated sequence of characters from the Universal Character Set [ISO10646].

The syntax of a UNC filespace selector string, as defined by [MS-DTYP], is given here in Augmented Backus-Naur Form (ABNF) [RFC5234] for convenience. Note that this definition is informative only; the normative description is in [MS-DTYP].

```
UNC = "\\" hostname "\" sharename *( "\" objectname )
hostname   = netbios-name / fqdn / ip-address
sharename  = <name of share or resource to be accessed>
objectname = <depends on resource being accessed>
```

- "netbios-name" from [MS-NBTE], Section 2.2.1.
- "fqdn" from [RFC1035] or [RFC1123]
- "ip-address" from Section 2.1 of [RFC1123], or Section 2.2 of [RFC4291].

The precise format of "sharename" depends on the protocol; see: SMB [MS-SMB], NFS [RFC7530], NCP [NOVELL].

Appendix F. Collected Rules

Here are the collected syntax rules for all optional appendices, presented for convenience.

```
file-URI       = file-scheme ":" file-hier-part

file-scheme    = "file"

file-hier-part = "//" auth-path
               / local-path

auth-path      = [ file-auth ] path-absolute
               / [ file-auth ] file-absolute
               / unc-authority path-absolute

local-path     = [ drive-letter ] path-absolute
               / file-absolute

file-auth      = [ userinfo "@" ] host

unc-authority  = 2*3"/" [ userinfo "@" ] file-host

file-host      = inline-IP / IPv4address / reg-name

inline-IP      = "%5B" ( IPv6address / IPvFuture ) "%5D"

file-absolute  = "/" drive-letter path-absolute

drive-letter   = ALPHA ":"
               / ALPHA "|"
```

This collected syntax is intended to support file URIs that take the following forms:

Local files:

- "file:///path/to/file"

  A traditional file URI for a local file, with an empty authority.

- "file:/path/to/file"

  The minimal representation of a local file, with no authority field and an absolute path that begins with a slash "/".

- "file:c:/path/to/file"

The minimal representation of a local file in a DOS- or Windows-based environment, with no authority field and an absolute path that begins with a drive letter.

- "file:///c|/path/to/file"
- "file:/c|/path/to/file"
- "file:c|/path/to/file"

Regular DOS or Windows file URIs, with vertical line characters in the drive letter construct.

Non-local files:

- "file://host.example.com/path/to/file"

The representation of a non-local file, with an explicit authority.

- "file:////host.example.com/path/to/file"

The "traditional" representation of a non-local file, with an empty authority and a complete (transformed) UNC string in the path.

- "file://///host.example.com/path/to/file"

As above, with an extra slash between the empty authority and the transformed UNC string.

Author's Address

Matthew Kerwin
Queensland University of Technology
Victoria Park Road
Kelvin Grove, QLD 4059
Australia

Email: matthew.kerwin@qut.edu.au