Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: June 14, 2014

M. Kerwin
QUT

December 11, 2013

The 'file' URI Scheme
draft-kerwin-file-scheme-09

Abstract

This document specifies the file Uniform Resource Identifier (URI) scheme that was originally specified in RFC 1738. The purpose of this document is to keep the information about the scheme on standards track, since RFC 1738 has been made obsolete, and to promote interoperability by resolving disagreements between various implementations.

Note to Readers

This draft should be discussed on its github project page [github].

Status of This Memo

This Internet-Draft is submitted to IETF 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 June 14, 2014.

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

The 'file' URI scheme has historically had little or no interoperability between platforms. Further, implementers on a single platform have often disagreed on the syntax to use for a particular filesystem. This document attempts to resolve those problems, and define a standard scheme which is interoperable between different extant and future implementations. Additionally, it aims to ease implementation by conforming to a general syntax that allows existing URI parsing machinery to parse 'file' URIs.

URIs were previously defined in [RFC1738], which was updated by [RFC3986]. Those documents also specify how to define schemes for URIs.

The first definition for many URI schemes appeared in [RFC1738]. Because that document has been made obsolete, this document copies the 'file' URI scheme from it to allow that material to remain on standards track.

1.1. History

This section is non-normative.

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) [IANA-URI-Schemes]; however that definition omitted certain language included by former that clarified aspects such as:

- the use of slashes to donate 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.draft-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 basis of this document.

The 'file' URI scheme defined in [RFC1738] is referenced three times in the current URI Generic Syntax standard [RFC3986], despite the former's obsoletion:

1. Section 1.1 uses "file:///etc/hosts" as an example for identifying a resource in relation to the end-user's local context.

2. Section 1.2.3 mentions the "file" scheme regarding relative references.

3. Section 3.2.2 says that '...the "file" URI scheme is defined so that no authority, an empty host, and "localhost" all mean the end-user's machine...'.

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

1.2. Conventions and Terminology

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].

2. Scheme Definition

The 'file' URI scheme is used to identify and retrieve files accessible on a particular host computer, where a "file" is a named resource which can be accessed through the computer's filesystem interface. These file names are interpreted from the perspective of the user of a reference, rather than in relation to a globally-defined naming authority, so care ought to be taken when distributing 'file' URIs to ensure that such references are actually intended to be interpreted in relation to the end user's filesystem interface.

This scheme, unlike most other URI schemes, does not identify a resource that is universally accessible over the Internet.

The mechanism for retrieving a representation of a dereferenced 'file' URI is through the computer's filesystem interface; for example using the POSIX "open", "read" and "close" functions [POSIX].

Also note that 'file' and 'ftp' URIs are not the same, even when the target of the 'ftp' URI is the local host.

2.1. Components

The 'file' URI scheme conforms with the generic structure defined in [RFC3986], and can be described in terms of its components:

Scheme The literal value "file"

Authority The authority component of a 'file' URI describes the machine or system on which the file is accessible. If the authority refers to a remote system, from the point of view of the user of the URI, the implication is that the file system cannot be accessed, or perhaps that some other mechanism must be used to access the file. It does not imply that the file ought to be accessible over a network connection. No retrieval mechanism for files stored on a remote machine is defined by this specification.

The authority component is optional in a 'file' URI.

If present it is either: one of the special values "localhost" or the empty string (""); or the host name of the system on which the file is accessible.

If the authority component is omitted, or has either of the special values "localhost" or the empty string (""), it is interpreted as "the machine from which the URI is being interpreted".

Path The path component of a 'file' URI describes the hierarchical directory path to the file, using the slash ("/") character to separate directories. Implementations SHOULD translate between the slash-separated URI syntax and the local system's conventions for specifying file paths, where they differ. (See: Section 3.2)

Note that the leading slash, if any, is included in the path value. This is in accordance with the generic syntax provided in [RFC3986], but at odds with the definition of the "url-path" given in Section 3.1 of [RFC1738]. See discussions in Section 3 for the effect this has on Microsoft DOS and Windows drive letters, and on UNC file paths.

Some systems allow 'file' URIs to refer to directories. In this case, implementations MAY include a terminating slash character in the path, such as in:

```
file:///usr/local/bin/
```

The presence of a terminating slash character always indicates that the 'file' URI refers to a directory, but the absence of a slash does not necessarily indicate that it refers to a filesystem object other than a directory. Implementations MUST NOT include a trailing slash in any 'file' URIs they generate that do not refer to a directory, and ought to use other mechanisms to detect directories in any 'file' URIs they receive, if and when such detection is required.

Query The query component of a 'file' URI contains non-hierarchical data that, along with data in the path component, serves to identify a file. For example, in a versioning file system, the query component might be used to refer to a specific version of a file.

Few implementations are known to use or support query components in 'file' URIs.

Fragment The semantics of a fragment component are undefined by this specification. A protocol that employs 'file' URIs MAY define its

own semantics for fragment components in the context of that protocol.

Previous definitions of the 'file' URI scheme required two (or three) slashes between the scheme and path, so implementations that wish to remain interoperable with older implementations ought to include an authority component in any 'file' URIs they generate. See also: Section 3.1.

2.2. 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 borrowing the 'host', 'path-absolute' and 'segment' rules from [RFC3986] (as updated by [RFC6874]).

```
   fileURI       = "file" ":" ( auth-file / local-file )

   auth-file     = "//" ( host-file / nohost-file )

   host-file     = hostpart path-absolute
                                ;   file://<host>/<path>
                                ;   file://localhost/<path>

   nohost-file   = path-abs       ; begins with "/"
                 / path-abs-win   ; begins with drive-letter
                                ;   file:///<path>
                                ;   file:////<UNC-path>
                                ;   file://c:/<path> *

   local-file    = path-absolute  ;   file:/<path>
                 / path-abs-win   ;   file:c:/<path>

   hostpart      = "localhost" / host

   path-abs      = 1*( "/" segment )

   path-abs-win  = drive-letter path-absolute
   drive-letter  = ALPHA [ drive-marker ]
   drive-marker  = ":" / "|"
```

* The 'no-host-file' rule allows for dubious URIs that encode a Windows drive letter as the authority component. See: Section 3.4.

Note the difference between the "path-abs" and "path-absolute" rules: only the former allows a zero-length first segment followed by a slash, e.g. "//foo/bar"

Systems exhibit different levels of case-sensitivity. Implementations SHOULD 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 SHOULD NOT alter the case of 'file' URIs they transport.

3. Implementation Notes

3.1. Leading Slash

The historical definition of 'file' URIs stated that "... the "/" between the host (or port) and the url-path is NOT part of the url-path", [RFC1738], Section 3.1, and that the "... <host> can be ... the empty string", [RFC1738], Section 3.10.

The implication of this definition is that absolute file paths in a UNIX-like environment, when encoded as 'file' URIs, ought to have begin with "file:////". This rarely, if ever, eventuated, and historically 'file' URIs interpreted in UNIX-like environments have included the first slash as part of the file path. This is compatible with the updated generic syntax provided in [RFC3986].

In Microsoft DOS- and Windows-based systems the historical definition resulted in 'file' URIs of the form

```
file:///c:/path/to/file
```

This structure, with an empty host/authority, can be mapped directly to the the updated generic syntax provided in [RFC3986] by omitting the authority entirely; for example:

```
file:c:/path/to/file
```

However the same is not true for a URI with a non-empty authority. For example:

```
file://smb.example.com/c:/path/to/file
```

As such, and to maintain interoperability with existing implementations, and with any historically-generated static URIs, implementations likely to interact with Microsoft DOS and Windows file systems SHOULD ignore the leading slash in the path component of any 'file' URIs they receive, where an authority component is included (even if it is blank) and the first path segment is a drive letter.

See Section 3.4 below for discussion on recognising drive letters.

3.2. Hierarchical Structure

Most implementations of the 'file' URI scheme do a reasonable job of mapping the hierarchical part of a directory structure into the slash ("/") delimited hierarchy of the URI syntax, independent of the native platform's delimiter.

For example, on Microsoft Windows platforms, it is typical that the file system presents backslash ("\") as the file delimeter for file names, yet the URI's forward slash ("/") can be used in 'file' URIs interpreted on those platforms. Similarly, on (some) Macintosh OS versions, at least in some contexts, the colon (":") is used as the delimiter in the native presentation of file path names. Unix systems natively use the same forward slash ("/") delimiter for hierarchy, so there is a closer mapping between 'file' URI paths and native path names.

In accordance with Section 3.3 of [RFC3986], the path segments "." and "..", also known as dot-segments, are only interpreted within the URI path hierarchy and are removed as part of the resolution process ([RFC3986], Section 5.2). Implementations operating on or interacting with systems that allow dot-segments in their native path representation may be required to escape those segments using some other means when translating to and from 'file' URIs.

3.3. Absolute and relative file paths

The conventions for specifying absolute file paths differ from system to system. For example, in a UNIX-based system an absolute file path begins with a slash ("/") character, denoting the root of the filesystem, whereas on a Microsoft DOS- or Windows-based system an absolute file path begins with a drive letter (e.g. "c:\").

As relative references are resolved into their respective (absolute) target URIs according to Section 5 of [RFC3986], this document does not describe that resolution. However, a fully resolved URI may contain a non-absolute file path. For example, using a generic URI parser, the URI:

```
file:alpha/bravo/charlie
```

might be parsed and interpreted as: file 'charlie', in directory 'bravo', in directory 'alpha', on the machine on which the URI is being interpreted (i.e. localhost); however there is no indication of the location of the directory 'alpha' on that machine. By convention an absolute file path would begin with a slash ("/") character on a Unix-based system, or a drive letter (e.g. "c:\") on a Microsoft Windows system, etc.

Resolution of non-absolute file paths is undefined by this specification. A protocol that employs 'file' URIs MAY define its own rules for resolution of relative file paths in 'file' URIs used in the context of that protocol.

3.4. Drive Letters

Historically drive letters have been mapped into the top of a 'file' URI in various ways. On systems running some versions of Microsoft Windows the drive letter may be specified with a colon (":") character, however sometimes the colon is replaced with a pipe ("|") character, and in some implementations the colon is omitted entirely. The three representations MAY be considered equivalent, and any implementation which could interact with a Microsoft Windows environment SHOULD interpret a single letter, optionally followed by a colon or pipe character, in the first segment of the path as a drive letter (see the "drive-letter" rule in Section 2.2). For example, the following URIs:

```
file:///c:/TMP/test.txt
file:///c|/TMP/test.txt
file:///c/TMP/test.txt
```

when interpreted on the same machine, would refer to the same file:

```
c:\TMP\test.txt
```

Implementations SHOULD use a colon (":") character to specify drive letters when generating URIs for Microsoft DOS- and Windows-based systems.

Note that the generic URI syntax in [RFC3986] dictates that "if a URI contains an authority component [even if it's a blank authority], then the path component must ... begin with a slash ("/") character." However some systems running some versions of Microsoft Windows are known to omit the slash before the drive letter, effectively replacing the URI's authority component with the drive specification; for example, "file://c:/TMP/test.txt". Implementations that are likely to encounter such a URI MAY interpret it as a drive letter, but SHOULD NOT generate such URIs.

3.5. UNC File Paths

The Microsoft Windows Universal Naming Convention (UNC) [MS-DTYP] defines a convention for specifying the location of resources such as shared files or devices, for example Windows shares accessed via the SMB/CIFS protocol [MS-SMB2]. The general structure of a UNC file path, given in Augmented Backus-Naur Form (ABNF) [RFC5234], is:

```
   UNC        = "\\" hostname "\" sharename *( "\" objectname )
   hostname   = <NetBIOS name, FQDN, or IP address of a server>
   sharename  = <name of a share or resource to be accessed>
   objectname = <the name of an object>
```

Note that this syntax description is non-normative.

There are two prevalent means of representing a UNC file path as a 'file' URI, and they differ subtly in their semantics.

The first representation of a UNC file path as a 'file' URI copies the UNC 'hostname' into the URI 'host' field, and the UNC 'sharename' and 'objectname's, concatenated with forward slash ("/") characters, into the 'path'. For example, the following UNC path:

```
   \\server.example.com\Share\path\to\file.doc
```

would be represented as a 'file' URI as:

```
   file://server.example.com/Share/path/to/file.doc
         \_________________/\_____________________/
              hostname      sharename+objectnames
```

The implication of this representation is that, because of the presence of a non-localhost authority, the file path is not accessible using the regular filesystem interface from the machine on which the URI is being interpreted. As noted in Section 2.1, this doesn't necessarily preclude that the file might be accessible through some other mechanism.

The 'file' URI scheme is unusual in that it does not specify an Internet protocol or access method for shared files; as such, its utility in network protocols between hosts is limited. Examples of file server protocols that do define such access methods include SMB/CIFS [MS-SMB2], NFS [RFC3530], and NCP [NOVELL].

The second representation translates the UNC file path entirely into the 'path' segment of a 'file' URI, including both leading slashes. For example, the UNC path given above would be represented as a 'file' URI as:

```
   file:////server.example.com/Share/path/to/file.doc
         \_____________________________________________/
                      translated UNC path
```

The implication of this representation is that the full UNC path can be accessed from the machine on which the URI is being interpreted using its regular filesystem interface.

3.5.1. Historical Issues with UNC File Paths

As mentioned in Section 3.1, the historical definition of 'file' URIs in [RFC1738] excluded the first slash ("/") character after the protocol identifier ("file://") from the file path. As such there exists a common variant of the second representation above, notably used by the Firefox web browser, that includes a fifth slash between the protocol identifier/authority and the UNC hostname. For example:

```
file://///server.example.com/Share/path/to/file.doc
       |\_____________________________________/
       |           translated UNC path
      extra slash
```

Implementations MAY interpret 'file' URIs with five slashes (three between a blank authority and the first non-empty path segment) as a UNC file path, but SHOULD NOT generate such URIs.

3.6. 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 [MS-DTYP] in Win32 File Namespaces, referred to as "Long UNC", using the prefix "\\?\UNC\".

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

4. Encoding and Character Set Considerations

As specified in [RFC3986], the 'file' URI scheme allows any character from the Universal Character Set (UCS) [ISO10646] encoded as UTF-8 [RFC3629] and then percent-encoded in valid ASCII [RFC20].

If the local file system uses a known non-Unicode character encoding, the file path SHOULD be converted to a sequence of Unicode characters normalized according to Normalization Form C (NFC, [UTR15]).

Before applying any percent-encoding, an application MUST ensure the following about the string that is used as input to the URI-construction process:

- The host, if any, consists only of Unicode code points that conform to the rules specified in [RFC5892].

- Internationalized domain name (IDN) labels are encoded as A-labels [RFC5890].

5. 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.

6. IANA Considerations

In accordance with the guidelines and registration procedures for new URI schemes [RFC4395], this section provides the information needed to update the registration of the 'file' URI scheme.

6.1. URI Scheme Name

file

6.2. Status

permanent

6.3. URI Scheme Syntax

See Section 2.2 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

6.4. URI Scheme Semantics

See Section 2 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

6.5. Encoding Considerations

See Section 4 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

6.6. Applications/Protocols That Use This URI Scheme Name

Web browsers:

- Firefox

  Note: Firefox has an interpretation of RFC 1738 which affects UNC paths. See: Section 3.5.1, Bugzilla#107540 [1]

- Chromium
- Internet Explorer
- Opera

Other applications/protocols:

- Windows API

  PathCreateFromUrl function [2], MSDN

  UrlCreateFromPath function [3], MSDN

- Perl LWP

These lists are non-exhaustive.

6.7. Interoperability Considerations

Due to the convoluted history of the 'file' URI scheme there a many, varied implementations in existence. Many have converged over time, forming a few kernels of closely-related functionality, and RFC XXXX attempts to accommodate such common functionality. [Note to RFC Editor: please replace XXXX with the number issued to this document.] However there will always be exceptions, and this fact is recognised.

6.8. Security Considerations

See Section 4 of RFC XXXX [Note to RFC Editor: please replace XXXX with the number issued to this document.]

6.9. Contact

Matthew Kerwin, matthew.kerwin@qut.edu.au

6.10. Author/Change Controller

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

6.11. References

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

[2] "PathCreateFromUrl function", MSDN, June 2013, <http://msdn.microsoft.com/en-us/library/windows/desktop/bb773581(v=vs.85).aspx>.

[3] "UrlCreateFromPath function", MSDN, June 2013, <http://msdn.microsoft.com/en-us/library/windows/desktop/bb773773(v=vs.85).aspx>

7. Acknowledgements

This specification is derived from RFC 1738 [RFC1738], RFC 3986 [RFC3986], and I-D draft-hoffman-file-uri (expired) [I-D.draft-hoffman-file-uri]; the acknowledgements in those documents still apply.

8. References

8.1. Normative References

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

[RFC20] Cerf, V., "ASCII format for Network Interchange", RFC 20, October 1969.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, February 2013.

[UTR15] Davis, M. and K. Whistler, "Unicode Normalization Forms", August 2012, <http://www.unicode.org/reports/tr15/tr15-37.html>.

8.2. Informative References

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

[IANA-URI-Schemes] Internet Assigned Numbers Authority, "Uniform Resource Identifier (URI) Schemes registry", June 2013, <http://www.iana.org/assignments/uri-schemes/uri-schemes.xml>.

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

[MS-SMB2] Microsoft Open Specifications, "Server Message Block (SMB) Protocol Versions 2 and 3", January 2013, <http://msdn.microsoft.com/en-us/library/cc246482.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, <http://www.unix.org/version4/>.

[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, June 1994.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

[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, <http://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx#namespaces>.

[github] Kerwin, M., "file-uri-scheme GitHub repository", n.d., <https://github.com/phluid61/file-uri-scheme>.

Author's Address

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

EMail: matthew.kerwin@qut.edu.au