Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2013

M. Kerwin

June 27, 2013

# The file URI Scheme
# draft-kerwin-file-scheme-04

## Abstract

This document specifies the file Uniform Resource Identifier (URI) scheme that was originally specified in [RFC1738]. The purpose of this document is to keep the information about the scheme on standards track, since [RFC1738] has been made obsolete.

## 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 December 29, 2013.

Table of Contents



## 1. Introduction

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. 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. History

The file URI scheme was first defined in [RFC1630], an informational RFC which does not specify an Internet standard of any kind. The definition was standardised in [RFC1738], and the scheme was registered with the Internet Assigned Numbers Authority (IANA) [IANA-URI-Schemes]; however the latter 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

- the requirement that client software convert the file: URL into a file name in the local file name conventions

- a justification for defining the scheme.

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.

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

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.

## 3. Scheme Definition

The file URI scheme is used to designate files accessible on a particular host computer. This scheme, unlike most other URI schemes, does not designate a resource that is universally accessible over the Internet.

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 machinery to parse file: URIs.

Note that file: and ftp: URIs are not the same, even when the target of the ftp: URI is the local host.

The syntax of a file: URI conforms with the generic syntax presented in [RFC3986], with the following components:

scheme name
The literal value "file"

authority
If present, either the fully qualified domain name of the system on which the file is accessible; or one of the special values "localhost" or the empty string (""), in which case it is interpreted as "the machine from which the URI is being interpreted". An absent authority component SHOULD be interpreted as if it were present and had the value "localhost".

A host name, if supplied, is intended to inform a client on a remote machine that it cannot access the file system, or perhaps to use some other mechanism to access the file. It does not imply that the file should be accessible over a network connection.

path
The hierarchical directory path to the file, using the slash character ("/") to separate directories. Implementations SHOULD translate between the URI syntax and the local system's conventions for specifying file paths, where they differ. (See: Section 4.1)

Some systems allow file: URIs to point to directories. In this case, there is usually (but not always) a terminating slash character, such as in:

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

Because the file URI scheme does not define a retrieval mechanism for dereferencing a file: URI, the semantics of a query or fragment component are considered unknown and are effectively unconstrained. A protocol or system that utilises the file URI scheme MAY define its own semantics for query and/or fragment components for file: URIs it uses.

Systems exhibit different levels of case-sensitivity. Implementations SHOULD attempt to 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.

## 4. Implementation Notes

### 4.1. 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. 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 resolved native path representation may be required to escape those segments using some other means.

### 4.2. Relative file paths

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

```
file:a/b/c
```

would be interpreted as file "c", in directory "b", in directory "a", on the machine on which the URI is being interpreted (i.e. localhost); however there is no apparent indication of the location of the directory "a" 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 relative file paths is left undefined by this specification.

### 4.3. Drives, drive letters, mount points, file system root

Historically there has been considerable difference, in practice, for handling of the syntax for the "top" of the hierarchy. The file: URI syntax provides one simple place for designating the root of the file hierachy, and implementations have diverged, even on the same platform, sometimes even within a single application.

For example, Microsoft DOS- and Windows-based systems support the notion of a "drive letter", a single character which represents a (virtual) drive, mount point, or device. Native representations of file paths start with the drive letter, a colon, and then the path; e.g., "c:\TMP\test.txt".

Drive letters are 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. 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.

Note that some systems running some versions of Microsoft Windows are known to omit the slash before the drive letter, effectively replacing the authority component with the drive specification. In line with Postel's robustness principle ("an implementation must be conservative in its sending behavior, and liberal in its receiving behavior" [RFC791]) implementations that are likely to encounter such a URI MAY interpret it as a drive letter, but SHOULD NOT generate such URIs.

### 4.4. UNC File Paths

File names encoded using the Universal Naming Convention (UNC) [MS-DTYP], for example Windows shares accessed via the SMB/CIFS protocol [MS-SMB2], SHOULD be translated entirely into the path segment of a file: URI, including both leading slashes. For example, the UNC path

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

would become

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

According to [RFC3986], a URI that does not contain an authority component cannot begin with two slash characters ("//"). Therefore a file: URI that includes a UNC path MUST include an authority component.

Note that the "hostname" part of a UNC path refers to the server or domain hosting the shared resource, and is usually different from the "host" part of the file: URI, which describes the machine from which the UNC hostname can be resolved.

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.

### 4.5. 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, using the prefix _\\?\UNC_.

This document does not define a mechanism for encoding namespaced file paths into file: URIs.

### 4.6. Character sets and encodings

Local file systems sometimes use many different encodings for representing file names. The URI syntax defined in [RFC3986] provides a method of encoding data, presumably for the sake of identifying a resource, as a sequence of characters. The URI

characters are, in turn, frequently encoded as octets for transport or presentation. This specification does not mandate any particular character encoding for mapping between URI characters and the octets used to store or transmit those characters, however for the sake of interoperability, file: URI libraries MAY translate the native character encoding for file names to and from their equivalent Unicode representation [UNICODE] encoded as UTF-8 [RFC3629] and then percent-encoded into valid ASCII [RFC20].

A protocol or system that utilises the file URI scheme MAY restrict the encoding of file: URIs it uses, and SHOULD declare such restrictions. If no such declaration is given, implementations SHOULD expect percent-encoded UTF-8 Unicode, as described above.

## 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. Under no circumstance should software using file: URIs grant greater access than would be available for other file access methods.

## 6. IANA Considerations

This document does not modify the existing entry in the URI Schemes registry [IANA-URI-Schemes], except by updating its reference RFC.

## 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

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

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

## 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.

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

[RFC791] Postel, J., "Internet Protocol - DARPA Internet Program, Protocol Specification", RFC 791, September 1981.

[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.

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

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

[MS-DTYP] Microsoft Open Specifications, "Windows Data Types, Section 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>.

[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>.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.1", 2012, <http://www.unicode.org/versions/Unicode6.1.0/>.

[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

EMail: matthew@kerwin.net.au