Applications Area Working Group   M. Kerwin
Internet-Draft   QUT
Updates: 1738 (if approved)   December 16, 2016
Intended status: Standards Track
Expires: June 19, 2017

                          The file URI Scheme
                   draft-ietf-appsawg-file-scheme-16

Abstract

   This document provides a more complete specification of the "file"
   Uniform Resource Identifier (URI) scheme, replacing the very brief
   definition in Section 3.10 of RFC 1738.

   It defines a common syntax which is intended to interoperate across
   the broad spectrum of existing usages. At the same time it notes
   some other current practices around the use of file URIs.

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 June 19, 2017.

Table of Contents

1. Introduction

A file URI identifies an object (a "file") stored in a structured
object naming and accessing environment on a host (a "file system.")
The URI can be used in discussions about the file, and if other

conditions are met it can be dereferenced to directly access the
file.

This document specifies a syntax based on the generic syntax of
[RFC3986] that is compatible with most existing usages. Where
incompatibilities arise they are usually in parts of the scheme that
were underspecified in earlier definitions and have been tightened up
by more recent specifications. Appendix A lists significant changes
to syntax.

Extensions to the syntax which might be encountered in practice are
listed in Appendix E; these extensions are listed for informational
purposes and are not a requirement of implementation.

The file URI scheme is not coupled with a specific protocol, nor with
a specific media type [RFC6838]. See Section 3 for a discussion of
operations that can be performed on the object identified by a file
URI.

1.1. 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] when they
appear in all upper case. They may also appear in lower or mixed
case as English words, without normative meaning.

Throughout this document the term "local file" is used to describe
files that can be accessed through the local file system API using
only the information included in the file path, not relying on other
information (such as network addresses.) 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.

The term "local file URI" is used to describe file URIs which have no
"authority" component, or where the authority is the special string
"localhost" or a fully qualified domain name that resolves to the
machine from which the URI is being interpreted (Section 2).

2. Syntax

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

The generic syntax in [RFC3986] includes "path" and "authority"
components, for each of which only a subset is used in the definition

of the file URI scheme. The relevant subset of "path" is "path-
absolute", and the subset of "authority" is "file-auth", given below.

The syntax definition below is different from those given in
[RFC1630] and [RFC1738] as it is derived from the generic syntax of
[RFC3986], which post-dates the previous file URI specifications.
Appendix A enumerates significant differences.

```
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      = "localhost"
                 / host
```

The "host" is the fully qualified domain name of the system on which
the file is accessible. This allows a client on another system to
know that it cannot access the file system, or perhaps that it needs
to use some other local mechanism to access the file.

As a special case, the "file-auth" rule can match the string
"localhost" which is interpreted as "the machine from which the URI
is being interpreted," exactly as if no authority were present. Some
current usages of the scheme incorrectly interpret all values in the
authority of a file URI, including "localhost", as non-local. Yet
others interpret any value as local, even if the "host" does not
resolve to the local machine. To maximize compatibility with
previous specifications, users MAY choose to include an "auth-path"
with no "file-auth" when creating a URI.

The path component represents the absolute path to the file in the
file system. See Appendix D for some discussion of system-specific
concerns including absolute file paths and file system roots.

Some file systems have case-sensitive file naming and some do not.
As such the file URI scheme supports case sensitivity, in order to
retain the case as given. Any transport-related handling of the file
URI scheme MUST retain the case as given. Any mapping to or from a
case-insensitive form is solely the responsibility of the
implementation processing the file URI on behalf of the referenced
file system.

Also see Appendix E that lists some nonstandard syntax variations
that can be encountered in practice.

3. Operations Involving file URIs

See the POSIX file and directory operations [POSIX] for examples of
standardized operations that can be performed on files.

A file URI can be dependably dereferenced or translated to a local
file path only if it is local. A file URI is considered "local" if
it has no "file-auth", or the "file-auth" is the special string
"localhost" or a fully qualified domain name that resolves to the
machine from which the URI is being interpreted (Section 2).

This specification neither defines nor forbids any set of operations
that might be performed on a file identified by a non-local file URI.

4. File System Name Encoding

File systems use various encoding schemes to store file and directory
names. Many modern file systems store file and directory names as
arbitrary sequences of octets, in which case the representation as an
encoded string often depends on the user's localization settings, or
defaults to UTF-8 [STD63].

When a file URI is produced that represents textual data consisting
of characters from the Unicode Standard coded character set
[UNICODE], the data SHOULD be encoded as octets according to the
UTF-8 character encoding scheme [STD63] before percent-encoding is
applied; as per [RFC3986], Section 2.5.

A decision not to use percent-encoded UTF-8 is outside the scope of
this specification. It will typically require the use of heuristics
or explicit knowledge about the way the string will be processed.

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.

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 as the origin for
each file URI [RFC6454], which is the most secure option.

Treating a non-local file URI as local or otherwise attempting to
perform local operations on a non-local URI can result in security
problems.

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 not be possible for this
specification to list all such significant characters and device
names. Implementers should 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.

File systems vary in the way they handle case. Care must be taken to
avoid issues resulting from possibly unexpected aliasing from case-
only differences between file paths or URIs, or from mismatched
encodings or Unicode equivalences [UAX15] (see Section 4).

6. 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.

   Used in development libraries, such as:

   * Windows Shell (PathCreateFromUrl, UrlCreateFromPath).

   * libwww-perl - The World-Wide Web library for Perl.

Contact:
   Applications and Real-Time Area <art@ietf.org>

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

References:
   This RFC.

7. Acknowledgements

Contributions from many members of the IETF and W3C communities -
notably Dave Crocker, Graham Klyne, Tom Petch, and John Klensin - are
greatly appreciated.

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 early comments and
suggestions; and to Paul Hoffman, whose earlier work served as an
inspiration for this undertaking.

8. References

8.1. Normative References

[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>.

[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>.

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

   [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>.

   [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>.

8.2. Informative References

   [Bash-Tilde]
              Free Software Foundation, Inc, "Bash Reference Manual:
              Tilde Expansion", February 2014,
              <http://www.gnu.org/software/bash/manual/html_node/
              Tilde-Expansion.html>.

   [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>.

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

   [MS-DTYP] Microsoft Open Specifications, "Windows Data Types, 2.2.57
              UNC", October 2015,
              <http://msdn.microsoft.com/en-us/library/gg465305.aspx>.

   [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>.

   [RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifiers (URI): Generic Syntax", RFC 2396,
              DOI 10.17487/RFC2396, August 1998,
              <http://www.rfc-editor.org/info/rfc2396>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
           Specifications and Registration Procedures", BCP 13,
           RFC 6838, DOI 10.17487/RFC6838, January 2013,
           <http://www.rfc-editor.org/info/rfc6838>.

[UAX15]   Davis, M., Ed. and K. Whistler, Ed., "Unicode Standard
           Annex #15: Unicode Normalization Forms", February 2016,
           <http://www.unicode.org/reports/tr15/tr15-44.html>.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version
           9.0.0", ISBN 978-1-936213-13-9, June 2016,
           <http://www.unicode.org/versions/Unicode9.0.0/>.

[WHATWG-URL]
           WHATWG, "URL Standard", December 2016,
           <https://url.spec.whatwg.org/>.

[Win32-Namespaces]
           Microsoft Developer Network, "Naming Files, Paths, and
           Namespaces", June 2013, <https://msdn.microsoft.com/en-
           us/library/windows/desktop/aa365247(v=vs.85).aspx>.

[Zsh-Tilde]
           "zsh: 14.7 Filename Expansion", December 2015,
           <http://zsh.sourceforge.net/Doc/Release/
           Expansion.html#Filename-Expansion>.

Appendix A. Differences from Previous Specifications

   The syntax definition in Section 2 inherits incremental differences
   from the general syntax of [RFC1738] made by [RFC2396] ([RFC2396],
   Appendix G) and [RFC3986] ([RFC3986], Appendix D).

   According to the definition in [RFC1738] a file URL always started
   with the token "file://", followed by an (optionally blank) host name
   and a "/". The syntax given in Section 2 makes the entire authority
   component, including the double slashes "//", optional.

Appendix B. Example URIs

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

   Local files:

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

        * "file:///path/to/file"

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

        * "file:/path/to/file"

   Non-local files:

   o A non-local file, with an explicit authority. E.g.:

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

Appendix C. Similar Technologies

   o The WHATWG URL specification [WHATWG-URL] defines browser behavior
      for a variety of inputs, including file URIs. As a living
      document, it changes to reflect updates in browser behavior. As a
      result, its algorithms and syntax definitions may or may not be
      consistent with this specification. Implementors should be aware
      of this possible discrepancy if they expect to share file URIs
      with browsers that follow the WHATWG specification.

   o 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, except that files located by UNC
      filespace selector strings are typically stored on a remote
      machine and accessed using a network protocol. Appendix E.3 lists
      some ways in which UNC filespace selector strings are currently
      made to interoperate with the file URI scheme.

   o 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.

Appendix D. 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. This is not an exhaustive list of operating or
   file systems; rather it is intended to illustrate certain types of
   interactions that might be encountered.

D.1. POSIX Systems

In a POSIX file system the root of the file system is represented as
a directory with a zero-length name, usually written as "/"; the
presence of this root in a file URI can be taken as given by the
initial slash in the "path-absolute" rule.

Common UNIX shells such as the Bourne-Again SHell (bash) and Z Shell
(zsh) provide a function known as "tilde expansion" [Bash-Tilde] or
"filename expansion" [Zsh-Tilde], where a path that begins with a
tilde character "~" can be expanded out to a special directory name.
No such facility exists using the file URI scheme; a tilde in a file
URI is always just a tilde.

D.2. DOS- and Windows-Like Systems

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

Appendix E lists some nonstandard techniques for interacting with
DOS- or Windows-like file paths and URIs.

D.3. Mac OS X Systems

The HFS+ file system uses a nonstandard normalization form, similar
to Normalization Form D [UAX15]. Take care when transforming HFS+
file paths to and from URIs (Section 4).

D.4. OpenVMS Files-11 Systems

When mapping a VMS file path to a file URI, the device name is mapped
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, that reference is
used as the authority. Where the original node reference includes a
user name and password in an access control string, they can be
transcribed into the authority using the nonstandard syntax extension
in Appendix E.1.

Appendix E. Nonstandard Syntax Variations

These variations may be encountered by existing usages of the file
URI scheme, but are not supported by the normative syntax of
Section 2.

This appendix is not normative.

E.1. User Information

It might be necessary to include user information such as a user name
in a file URI, for example when mapping a VMS file path with a node
reference that includes an access control string.

To allow user information to be included in a file URI, the "file-
auth" rule in Section 2 can be replaced with the following:

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

This uses the "userinfo" rule from [RFC3986].

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]. Take care when dealing with information that can be used
to identify a user or grant access to a system.

E.2. DOS and Windows Drive Letters

On Windows- or DOS-like file systems an 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 ":"
```

The "ALPHA" rule is defined in [RFC5234].

This is intended to support the minimal representation of a local
file in a DOS- or Windows-like environment, with no authority field
and an absolute path that begins with a drive letter. E.g.:

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

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. In some usages of file URIs drive letters are

   canonicalized by converting them to uppercase, and other usages treat
   URIs that differ only in the case of the drive letter as identical.

E.2.1. Relative Resolution

   To mimic the behaviour of DOS- or Windows-like file systems, relative
   references beginning with a slash "/" can be resolved relative to the
   drive letter, when present; and resolution of ".." dot segments (per
   Section 5.2.4 of [RFC3986]) can be modified to not ever overwrite the
   drive letter.

   For example:

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

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

   A relative reference starting with a drive letter would be
   interpreted by a generic URI parser as a URI with the drive letter as
   its scheme. Instead such a reference ought to be constructed with a
   leading slash "/" character (e.g. "/c:/foo.txt").

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

E.2.2. Vertical Line Character

   Historically some usages of file URIs have included 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 can be 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 regular DOS or Windows file URIs with vertical line characters in the drive letter construct. E.g.:

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

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

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

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

[E.3](#). UNC Strings

Some usages of the file URI scheme allow UNC filespace selector strings [[MS-DTYP](#)] to be translated to and from file URIs, either by mapping the equivalent segments of the two schemes (hostname to authority, sharename+objectnames to path), or by mapping the entire UNC string to the path segment of a URI.

[E.3.1](#). file URI with Authority

The following is an algorithmic description of the process of translating a UNC filespace selector string to a file URI by mapping the equivalent segments of the two schemes:

1. Initialize the URI with the "file:" scheme identifier.

2. Append the authority:

    1. Append the "//" authority sigil to the URI.

    2. Append the host-name field of the UNC string to the URI.

3. Append the share-name:

1. Transform the share-name to a path segment ([RFC3986], Section 3.3) to conform to the encoding rules of Section 2 of [RFC3986].

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

4. For each object-name:

    1. Transform the objectname to a path segment as above.

        The colon character ":" is allowed as a delimiter before stream-name and stream-type in the file-name, if present.

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

For example:

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

The inverse algorithm, for translating a file URI to a UNC filespace selector string, is left as an exercise for the reader.

E.3.2. file URI with UNC Path

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 can be replaced with the following:

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

    unc-authority = 2*3"/" file-host

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

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

This syntax uses the "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 B:

Non-local files:

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

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

o As above, with an extra slash between the empty authority and the
   transformed UNC string, as per the syntax defined in [RFC1738].
   E.g.:

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

   This representation is notably used by the Firefox web browser.
   See Bugzilla#107540 [Bug107540].

It also further limits the definition of a "local file URI" by
excluding any file URI with a path that encodes a UNC string.

E.4. Backslash as Separator

Historically some usages have copied entire file paths into the path
components of file URIs. Where DOS or Windows file paths were thus
copied the 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 F. Collected Nonstandard Rules

Here are the collected syntax rules for all optional appendices,
presented for convenience. This collected syntax is not normative.

```
      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      = "localhost"
                     / [ userinfo "@" ] host

      unc-authority  = 2*3"/" 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:

   o A traditional file URI for a local file, with an empty authority.
      E.g.:

      * "file:///path/to/file"

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

      * "file:/path/to/file"

   o 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. E.g.:

      * "file:c:/path/to/file"

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

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

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

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

   Non-local files:

   o The representation of a non-local file, with an explicit
      authority. E.g.:

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

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

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

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

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

Author's Address

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

   Email: matthew.kerwin@qut.edu.au