The ‘ftp’ URI Scheme

draft-yevstifeyev-ftp-uri-scheme-05

Abstract

This document specifies the ‘ftp’ Uniform Resource Identifier (URI) scheme, which is used to refer to resources accessible via File Transfer Protocol (FTP). It updates RFC 959 and RFC 1738.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2011 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
Table of Contents

1. Introduction .................................................. 3
2. Terminology and Conventions .................................. 3
  2.1. Conformance Criteria ..................................... 3
  2.2. Terminology ................................................ 3
  2.3. Formal Syntax .............................................. 4
  2.4. Interpreting Examples ..................................... 4
  2.5. Miscellaneous Conventions ................................ 4
3. URI Scheme Specification ......................................... 5
  3.1. URI Scheme Syntax .......................................... 5
  3.2. URI Scheme Semantics ....................................... 6
    3.2.1. The <host-port> Part ................................. 7
    3.2.2. The <user-pass> Part ................................ 7
    3.2.3. The <ftp-path> Part ................................ 9
      3.2.3.1. The <typecode-part> Part ...................... 10
    3.2.4. Queries and Fragment Identifiers .................. 11
      3.2.4.1. Queries ......................................... 11
      3.2.4.2. Fragment Identifiers .......................... 11
  3.3. Encoding Considerations .................................... 11
4. Examples ........................................................ 12
  4.1. Examples of 'ftp' IRIs and Internalized URIs ............... 16
5. Security and Privacy Considerations ............................ 17
6. Internalization Considerations .................................. 18
  6.1. UCS Characters in 'ftp' URIs .............................. 18
  6.2. 'ftp' IRIs ................................................ 18
    6.2.1. Mapping 'ftp' IRIs to 'ftp' URIs .................... 18
  6.3. Handling 'ftp' URIs with UCS Characters and 'ftp' IRIs .... 19
    6.3.1. Internalized <host> Part in 'ftp' URIs and <ihost> Part in 'ftp' URIs ..................................... 19
    6.3.2. Internalized <ftp-path> Part in 'ftp' URIs and <iftp-path> in 'ftp' IRIs .................................. 19
  6.4. Internalization of Actual Data Interchange .................. 20
7. IANA Considerations ............................................. 20
  7.1. The 'ftp' URI Scheme ...................................... 20
  7.2. The 'ftps' URI Scheme .................................... 20
  7.3. Maintaining ftp.uri.arpa Domain ........................... 21
8. References ....................................................... 21
  8.1. Normative References ..................................... 21
  8.2. Informative References ................................... 22
Appendix A. Dynamic Delegation Discovery System (DDDS) and 'ftp' URIs ........................................... 25
Appendix B. Previous Syntax Definitions ............................ 26
  B.1. RFC 1630 .................................................. 26
  B.2. RFC 1738 .................................................. 27
Appendix C. List of Changes since RFC 1738 ........................ 29
1. Introduction

File Transfer Protocol (FTP) is a standard network protocol used to copy a file from one host to another over a TCP-based network. It has had a very long history; the protocol is rooted in the early 1970s, the times of ARPANET, with the first specification being RFC 354 [RFC0354]; the most current FTP specification is RFC 959 [RFC0959]. RFC 1123 [RFC1123] made a number of changes to FTP specification.

The 'ftp' Uniform Resource Identifier (URI) scheme, used for referencing resources accessible via FTP, has been deployed. It was first mentioned in RFC 1630 [RFC1630] - pre-Standard Track RFC on URIs. Later, RFC 1738 [RFC1738] (which was the official specification of Uniform Resource Locators (URLs); see RFC 3305 [RFC3305]), Section 3.2 specified this scheme, as well as many others, on IETF Standards Track. This document extracts the definition of the 'ftp' URI scheme from this document to retain on Standard Track if RFC 1738 is moved to Historic status [RFC2026][HISTORIC] as well as makes several changes to suit current scheme usage. (With the first respect it belongs to a series of similar documents like RFC 2368 [RFC2368], which is now obsoleted by RFC 6068 [RFC6068], RFC 4248 [RFC4248], RFC 4266 [RFC4266], and RFC 5538 [RFC5538]; RFC 4156 [RFC4156] and RFC 4157 [RFC4157] also extracted definition of 'wais' and 'prospero' schemes from RFC 1738 but have no relation to Standards Track, since the aforementioned schemes are historical.) It updates RFC 959 and RFC 1738.

Generic URI syntax is described in RFC 3986 [RFC3986]; registration procedures for new URI schemes - in RFC 4395 [RFC4395].

2. Terminology and Conventions

2.1. Conformance Criteria

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. When these keywords are used with particular ABNF productions [RFC5234], they are meant to clarify syntactical requirements (e.g. "<foo> is OPTIONAL" means that one may safely omit <foo> in the URI).

2.2. Terminology
In this document, the terms "client" and "server" are used in the meaning of "user-FTP process" and "server-FTP process", respectively, which are defined in Section 2.2 of RFC 959 [RFC0959]. The terms "FTP command" (referred to as "command" within this document), "user-PI", "server-PI", "user-DTP", "server-DTP", "control connection", "data connection", "reply" and "user" are used with the meaning defined ibidem. Section 3.3 makes use of terms described in RFC 3536bis [RFC3536bis]. Terms related to DDDS used in Appendix A, especially those which occur capitalized, are described in RFC 3402 [RFC3402]. IDNA-related terminology is derived from RFC 5890 [RFC5890].

In this document "ASCII" refers to the American Standard Code for Information Interchange, defined in [ASCII] and amended for use in network interchange by RFC 20 [RFC0020]; definition of "ASCII" found in RFC 959 [RFC0959] may be considered to be equivalent.

2.3. Formal Syntax

This document uses the Augmented Backus-Naur Form (ABNF) [RFC5234] for description of formal syntax. The <host>, <port>, <unreserved>, <pct-encoded>, and <sub-delims> rules are imported from RFC 3986 [RFC3986] and <ALPHA> rule - from RFC 5234 [RFC5234].

2.4. Interpreting Examples

In the examples of FTP dialogs presented in this document, lines that begin "C> " were sent over the control connection from the user-PI to the server-PI, and lines that begin "S> " were sent over the control connection from the server-PI to the user-PI. In all cases, the prefixes shown above, including the one space, have been added for the purposes of this document, and are not a part of the data exchanged between client and server. Within such dialogs text enclosed in angle brackets ("<" and ">"), ASCII characters 0x3C and 0x3E, respectively) is not an actual part of FTP exchange but rather describes actions taken by parties of exchange or provides general comment.

2.5. Miscellaneous Conventions

The construction "ASCII character 0xHH", where "HH" represents 2 hexadecimal digits, is equivalent to RFC 20 construction "ASCII X’HH’" and denotes ASCII character which has been assigned the ASCII code HH. For example, ASCII character 0x5E refers to the "^" (caret) and ASCII character 0x7B refers to "{" (left curly bracket).

The constructions described in Sections 3 and 4 of RFC 5137 [RFC5137] are used in this document to point and escape Unicode characters,
respectively.

3. URI Scheme Specification

3.1. URI Scheme Syntax

The syntax of 'ftp' URI is described in <ftp-uri> rule below.

```
ftp-uri       = "ftp:" ftp-hier-part
ftp-hier-part = "//" [ user-pass "@" ] host-port [ ftp-path ]
user-pass     = user [ ":" pass ]
user          = 1*usp-char
pass          = *usp-char
usp-char      = unreserved / pct-encoded / sub-delims
host-port     = host [ ":" port ]
ftp-path      = [ cwd-part ] "/" last-segment [ typecode-part ]
cwd-part     = *( "/" cwd )
cwd          = segment-nsc
last-segment = segment-nsc
segment-nsc  = *pchar-nsc
pchar-nsc    = unreserved / pct-encoded / sub-delims-nsc / ":" / ":" / ":" / ":" / ":" / ":" / ":" ;
sub-delims-nsc = "!" / "$" / "&" / "/" / "(" / ")" / ":" / ":" / ":" / ":" / ":" / ":" ;
  RFC 3986 <sub-delims> excluding semicolon (";"")
typecode-part = ";;type=" typecode
  typecode      = "a" / "e" / "i" / "u" / "d" / typecode-ext
  typecode-ext  = ALPHA
```

RFC 3986 deprecated the use of "user:pass" format of the <userinfo> part of URIs. However, for some historical reasons, the benefits of the use of such construction for denoting the user information in the 'ftp' URIs are valuable enough to overlook this issue; see Section 3.2.2 of this document.

When <ftp-path> is present, it should be noted that <last-segment> is always present, too; it may be null, though. For instance, the URIs 
<ftp://exmaple.org/> and <ftp://example.net/foo/> have null <last-segment>s while <ftp://exmaple.com/big.xls> has the <last-segment> equal to "big.xls".

Please note that while processing the 'ftp' URI those characters which appear percent-encoded, MUST be decoded for the purpose of handling the URI, including the actual FTP exchange; see Section 3.3...
for more information.

The semantics of each part are defined in Section 3.2.

3.2. URI Scheme Semantics

The ‘ftp’ URI specifies a resource (a file or a directory listing) on the definite FTP server.

The application resolving the ‘ftp’ URI SHALL use the following algorithm:

1. establish the Transmission Control Protocol (TCP) [RFC0793] connection to the server identified by the <host> on the port identified by the <port> (or 21, if not supplied in the ‘ftp’ URI);

2. perform an attempt to identify the host it is trying to access using the HOST command [I-D ietf-ftpext2-hosts], as described in Section 3.2.1;

3. authenticate itself to the server;

4. request a list of supported features from server using FEAT command [RFC2389]; if feature negotiation mechanism is not supported by the server, act if the FEAT command has not been sent (this step is RECOMMENDED but not required); and

5. perform a series of commands according to <ftp-path> part.

Please note that the client MAY also perform other steps during this algorithm, such as requesting the server information using SYST command [RFC0959] or select a language of interchange using LANG command [RFC2640]. However, performing the steps of this algorithm is REQUIRED, modulo step 4, which is RECOMMENDED.

Handling error replies received during processing the URI, unless clearly stated in this document, is implementation-specific.

Since ‘ftp’ URI does not denote the transmission mode which is to be used, the stream mode, which is described in Section 3.4.1 of RFC 959 [RFC0959], MUST always be used.

‘ftp’ URIs cannot be used for other operations, such as uploading or removing a file on a server.

Note: The ‘ftp’ URI scheme supports FTP over TCP only; such derivations as FTP over User Datagram Protocol (UDP) [RFC0768] or
Stream Control Transmission Protocol (SCTP) [RFC4960], as known to be deployed, are not supported by it.

Note: The ’ftp’ and the ’file’ URI are not the same, even though they both may refer to the resource on the local host.

More detailed description of each URI’s parts’ semantics is below.

3.2.1. The <host-port> Part

The <host-port> part, which is REQUIRED, consists of the <host> and the <port> parts. The <host> part, which is REQUIRED within <host-port>, specifies the server which a connection is to be established to. The <port> part, which is OPTIONAL within <host-port>, denotes the TCP port for establishing such connection.

If the <port> part with the preceding colon (":") character (ASCII character 0x3A) is omitted, the port SHALL default to 21, as registered in [IANA-PORTREG].

Upon establishing a successful TCP connection, the client SHALL first try to identify the host it is trying to access using the HOST command [I-D ietf-ftpext2-hosts]. It is performed by sending this command with the <host> part of the URI as an argument.

If either 500 or 502 reply is received in response (which identify that the HOST command is unrecognized or unimplemented, respectively), the client SHALL act as if a HOST command had not been sent and continue processing the URI. If either 501 or 504 reply is received (which identify that the supplied hostname is syntactically invalid or it is unavailable, respectively), the client’s behavior depends on how does the server react. If, in accordance with Section 3.3 of RFC nnnn [I-D ietf-ftpext2-hosts], the server chooses to terminate the connection, the client SHALL notify the user and take no further actions. Otherwise, if the server does not terminate the connection, the client SHALL act as if a HOST command had not been sent and continue processing the URI.

3.2.2. The <user-pass> Part

The <user-pass> part, which is OPTIONAL, consists of the <user> and the <pass> parts. The <user> part, which is REQUIRED within <user-pass>, denotes the user name; the <pass> part, which is OPTIONAL within <user-pass>, - the password. The user name and the password are delimited by the colon (":") character (ASCII character 0x3A).

There are three cases of handling the <user-pass> part. The first implies that the ’ftp’ URI provides entire user credentials (a user
name and a password). In this case, upon establishing successful TCP connection to the server specified in the URI the client SHALL use supplied user name with the USER command; if the server requests the password via sending the 331 reply, one supplied in the URI SHALL first be used.

The second case covers the situation when the only user name is supplied. Under such circumstances, the client SHALL first use it in the USER command; if the server requests password, it SHALL be prompted from the user and then supplied with the PASS command.

The third case is when the whole <user-pass> part is omitted in the URI. In this case upon establishing the connection the "anonymous FTP" [RFC1635] SHALL be used; it implies use of the following credentials:

1. the user name "anonymous", and
2. the password "guest" or that which is the email address [RFC5322] of the user.

Note: Current FTP implementations mostly pay no attention to the password supplied with the "anonymous" user name. Thus clients SHOULD NOT supply real email addresses, due to the security reasons, but SHOULD rather supply either randomly-generated or non-existing email addresses under such circumstances. See also Section 5.

However, the authentication which implies use of <user-pass> part of the URI might be unsuccessful (ie. the server might fail to authenticate the user), which is indicated by receiving the 530 reply in response to either USER or PASS command. In this case, the user name and password SHALL be requested from the user and then used. If the credentials supplied by the user did not lead to successful authentication as well, they SHALL be requested once more unless and until the user gets authenticated or decides to terminate connection.

The ‘ftp’ URI does not provide a way to denote account information, used with ACCT command. Thus, if the server requests it for authentication (via sending 332 reply to a successful PASS command) or it is required for performing other command (which is denoted by either 332 or 532 server reply received upon sending such command), the handling of an URI SHALL be suspended, account information SHALL be requested from the user and then used prior to the continuation of URI processing. In the case when sever refuses to accept such account information, it SHALL be requested once more unless and until it is accepted by the server or the user decides to terminate connection.
The \(<user-pass>" part is not intended to define information which
should be used if the authentication is performed using the AUTH
command or other mechanism spelled out in RFC 2228 [RFC2228]; see
Section 5 of this document.

3.2.3. The \(<ftp-path>" Part

The \(<ftp-path>" part, which is OPTIONAL, denotes the resource (a file
or a directory listing) on the server specified by \(<host>". For
better understanding the algorithm below, the ABNF definition of
\(<ftp-path>" is copied here:

\[
\text{ftp-path} \quad = \quad [ \text{cwd-part} \quad /\" \quad /\" \quad \text{last-segment} \quad [ \text{typecode-part} \quad ] \quad ]
\]
\[
\text{cwd-part} \quad = \quad *( \text{"} /\" \text{cwd} \text{) } \]
\[
\text{cwd} \quad = \quad \text{segment-nsc}
\]
\[
\text{last-segment} \quad = \quad \text{segment-nsc}
\]
\[
\text{segment-nsc} \quad = \quad <\text{defined in Section 3.1}>\]
\[
\text{typecode-part} \quad = \quad \"\text{;type=}\text{typecode}\text{"
\]
\[
\text{typecode} \quad = \quad \"a\" \text{ / } \"e\" \text{ / } \"i\" \text{ / } \"u\" \text{ / } \"d\" \text{ / } \text{typecode-ext}
\]
\[
\text{typecode-ext} \quad = \quad \text{ALPHA}
\]

Please note that when \(<ftp-path>" is omitted, for the purpose of
processing the URI it MUST be considered to be \\
"/\" and SHOULD be
changed to \\
"/\" when normalizing the URI.

The \(<ftp-path>" part SHALL be processed using the following algorithm:

1) if the \(<cwd-part>" is present, each of \(<cwd> parts are
consistently supplied as arguments to the CWD (change working
directory) FTP command;

Note: Any null \(<cwd> parts, allowed per aforementioned syntax, MUST NOT cause sending CWD commands, since they might be erroneously interpreted by some FTP servers.

2) if the \(<typecode-part>" is present and \(<typecode> is either \\
"a", \\
"e", \\
"i" or \\
"u", perform the TYPE command with the \(<typecode> as
an argument;

3) whatever the path is, arrange the data connection to the server
using an appropriate method, as per those facilities of server
discovered with FEAT command [RFC2389] (eg. PORT, PASV
[RFC0959], EPRT or EPSV [RFC2428] command; using LPRT and LPSV
[RFC1639] commands, which were designated as historical by RFC 5797 [RFC5797], is strongly discouraged);

4a) if \(<last-segment>" is null (whatever \(<typecode> is), retrieve the
listing of current directory using appropriate method, as per
those facilities of server discovered with FEAT command (RFC2389) (eg. LIST, NLST [RFC0959] or MLSD [RFC3659] command);

Note: If <cwd-part> and <typecode-part> are omitted and <last-segment> is null, <ftp-path> refers to the default directory on the <host> for the logged user; in this case directory listings are to be retrieved directly after establishing data connection, skipping steps (1) and (2) above.

(4b) if the <typecode-part> is present and the <typecode> is equal to "d", <last-segment> specifies the directory; in this case retrieve listing of the directory specified by <last-segment> using the appropriate method (see above);

(4c) in the case described in (2) <last-segment> refers to a file; retrieve this file using appropriate method (using RETR command is RECOMMENDED);

(4d) otherwise, <last-segment> may refer either to a file or a directory listing; perform both subsequent attempts to access the file and a directory listing whereas the order of such attempts is non-substantial.

Note: Some client may involve additional heuristic algorithms to determine what does the <last-segment> refer to, such as checking its format to see whether is matches the "<name>.<extension>" format. This document allows them to do in such way.

Please note that the client MAY also perform other steps during this algorithm, such as retrieving file size using SIZE command or modification time using MSTM command [RFC3659]. However, performing the steps of this algorithm is REQUIRED.

If the reply which identifies the absence of the resource (a directory or a file) identified by one of the <cwd> parts or a <last-segment> part of the URI (explicitly, 550 reply) is received, the client SHALL stop processing the 'ftp' URI, remain the most currently accessed directory active, notify the user, and take no further actions.

Handling other error replies caused by processing the <ftp-path> is implementation-specific.

3.2.3.1. The <typecode-part> Part

The <typecode-part> part specifies the data type of <last-segment>. Currently, there are five options of <typecode>: "a", "e", "i", which are specified in RFC 959 [RFC0959] and stand for ASCII [ASCII],
The EBCDIC [RFC0183] text, "raw" (binary) data, "u", is specified in RFC mmmm [I-D ietf-ftpext2-typeu] and stands for Unicode [RFC5198] text, and "d", which is not an actual typecode but rather a "pseudo-typecode" to identify that the <last-segment> is a directory.

The <typecode-ext> production provides a possibility to accommodate new typecodes in the ‘ftp’ URI. Therefore, when a new FTP data type is defined, its specification MUST define its relationship with the ‘ftp’ URI.

3.2.4. Queries and Fragment Identifiers

3.2.4.1. Queries

This document does not specify the query component to be used in ‘ftp’ URIs. Clients MAY ignore it, if present. Correspondingly, any question mark ("?") characters (ASCII character 0x3F), as they are not allowed within URIs for any reason other that denoting the query component by RFC 3986, MUST be percent-encoded within ‘ftp’ URIs.

3.2.4.2. Fragment Identifiers

According to RFC 3986, the specification of a definite URI scheme must not define the fragment identifiers in the corresponding scheme syntax, as they depend on the media type of a resource identified by such URI. Correspondingly, fragment identifier are allowed in any URI.

The number sign ("#") characters (ASCII character 0x23), if used for the reason other than to delimit the fragment identifier SHALL be percent-encoded.

If present in ‘ftp’ URI, fragment identifier is put after the <ftp-path> (and <typecode-part>, if present), forming an URI like <ftp://example.org/file.txt;type=a#char=100>.

Since a number of clients prefer to save but do not open the file identified by the ‘ftp’ URI, they may be unable to process fragment identifier. Therefore, processing of fragment identifiers in ‘ftp’ URIs is REQUIRED when the file is actually processed by the client. This document does not specify the specific rules of handling fragment identifiers, since it is done by the documentation of specific ones, such as RFC 5147 [RFC5147].

See Section 4 for an example of an URI with fragment identifier.

3.3. Encoding Considerations
The parts of ‘ftp’ URIs may contain characters from ASCII character set which are allowed in the corresponding parts. Those characters which are excluded from the allowed characters for a particular part SHALL be encoded within this part.

‘ftp’ URIs MAY contain characters from Universal Character Set (UCS) [UCS] in <host> and <ftp-path> part, as discussed in Section 6. Further internalization of ‘ftp’ URIs is discussed ibidem.

As mentioned before, the characters in ASCII range which appear percent-encoded in the URI, MUST be decoded in the actual FTP exchange. This means that when sending data over FTP control connection per Section 3.2 of this document percent-encoded characters SHALL be replaced with their ASCII equivalents. However, those percent-encoded octets which are outside of ASCII range SHALL be transmitted as the corresponding octets. For instance, "%2F", if occurs in the <cwd>, will be replaced with "/", ASCII character 0x2F, and "%E7", if occurs ibidem, will be transmitted as octet <E7> when sending as an argument to CWD command.

4. Examples

This section provides several examples of ‘ftp’ URIs and their valid handling per this document. Within it, DNS names reserved by RFC 2606 [RFC2606] and IPv4 addresses reserved by RFC 5737 [RFC5737] are used.

The URI

<ftp://example.com:49557/%2Fsomedir/seconddir;type=d>

may result in the following data exchange:

    <client connecting to example.com on port 49557>
S> 220 ExampleFTP Server ready
C> HOST example.com
S> 220 Host accepted
C> USER anonymous
S> 331 Anonymous permitted; supply email as password
C> PASS bad-guy@example.com
S> 230 Logged in
C> CWD /somedir
S> 250 Directory changed
C> PASV
S> 227 Entering Passive Mode (192,0,2,12,52,453)
C> NLST seconddir
S> 150 Here comes the directory listing
     <server-DTP sending directory listing over data connection

Yevstifeyev             Expires January 26, 2012               [Page 12]
to user-DTP>
S> 226 Directory listing sent

The URI

<ftp://fellow:bad-guy@203.0.113.42/%2Fetc/motd?some=thing>

may result in the following data exchange:

<client connecting to 203.0.113.42 on port 21>
S> 220 CoolFTP Server Ready
C> HOST 203.0.113.42
S> 220 Host OK
C> USER fellow
S> 331 Specify password
C> PASS bad-guy
S> 230 Congrats! Logged in
C> CWD /etc
S> 250 Directory changed
C> PASV
S> 227 Passive entered (203,0,113,42,61,853)
<the <last-segment> is a directory to the client’s mind>
C> LIST motd
S> 550 No such directory
<in this case <last-segment> refers to a file>
C> RETR motd
S> 150 Transfer starts...
<server-DTP sending motd over data connection to user-DTP>
S> 226 File is sent
<client ignores query component>

Such URI is different from one with <ftp-path> equal to "/etc/motd"
or "///etc/motd", since both such URI will result in sending "CWD etc"instead of <CWD /etc>.

The following example illustrates the situation when suppliedcredentials are invalid. Thus, the URI

<ftp://user1:invalid-pass@example.net:4916;/type=d>

may result in the following:

<connecting to example.net on port 4916>
S> 220 GigaSoft FTP - welcome
C> HOST example.net
S> 220 Why not :-) 
C> USER user1
S> 331 Mention password
C> PASS invalid-pass
S> 530 Invalid credentials
   <client requests credentials from user>
   <user specified: "right-user" as user name and "right-pass" as password>
C> USER right-user
S> 331 Mention password
C> PASS right-pass
S> 230 right-user is a cool guy
C> FEAT
S> 211-Here the listing comes
S>  AUTH TLS
S>  TYPE a;u
S>  MLST
S> 211 End
C> PASV
S> 227 Passive opened on (198,51,100,2,65,123)
C> MLSD
S> 150 Here comes the directory listing
   <server-DTP sending directory listing over data connection to user-DTP>
S> 226 Directory listing sent

The following URI contains percent-encoded "?" and "#" characters and fragment identifier to illustrate their valid handling. The URI:

<ftp://exmaple.org/%3Ffoo/%23bar/file.txt;type=a#char=500>

may result in the following data exchange:

   <client connecting to example.org on port 21>
S> 220 Hello
C> HOST exmaple.org
S> 220 OK
C> USER anonymous
S> 230 No pass required
C> CWD ?foo
S> 250 Directory changed
C> CWD #bar
S> 250 Directory changed
C> TYPE a
S> 200 Accepted
C> PORT 198,51,100,2,65,123
S> 200 Accepted
   <the <last-segment> refers to a file to the client’s mind>
C> RETR file.txt
S> 150 Start transmission
   <server-DTP sending directory listing over data connection
The last example illustrates the complicated URI where a number of issues should be considered. Such issues include the server refusing to accept host name with HOST command, invalid user credentials, inability to support the "u" TYPE parameter and the absence of "bad-file.doc". The URI:

<ftp://oh-no@example.org:7634/foo/bar/foobar/bad-file.doc;type=u>

may result in the following data interchange:

S> 220 Yevstifeyev FTP ready
C> HOST example.org
S> 504 Unknown host
<server does not close connection>
C> USER oh-no
S> 331 Supply password
<password not supplied in URI; client requests one from user>
C> PASS some-pass
S> 530 Invalid credentials
<client requests credentials from user>
C> USER cool-man
S> 331 Supply password
C> PASS cool-pass
S> 230 Authenticated
C> CWD foo
S> 250 foo is an active directory
C> CWD bar
S> 250 foo/bar is an active directory
C> CWD foobar
S> 250 foo/bar/foobar is an active directory
C> TYPE u
S> 504 U TYPE not supported
C> PORT 192,0,2,14,64,695
S> 200 Accepted
C> RETR bad-file.doc
S> 550 No such file :-(
<client notifies user>
4.1. Examples of 'ftp' IRIs and Internalized URIs

This section provides several examples of handling 'ftp' IRIs and internalized URIs, as defined in Section 6.

The IRI, which contains U+2603 SNOWMAN character in the <ftp-path> as one of <cwd>s and U+0109 LATIN SMALL LETTER C WITH CIRCUMFLEX character in the <host>,

ftp://&x0109;at.example.com/weather/☃/snow.txt

may result in the following data exchange:

   <client connecting to xn--at-0la.example.com on port 21>
   xn--at-0la.example.com is Punycode-converted <host> in the IRI>
S> 200 Hi, man!
C> HOST xn--at-0la.example.com
S> 220 That’s OK.
C> USER anonymous
S> 331 Supply password
C> PASS foo@example.org
S> 230 Logged in
C> FEAT
S> 211-Features listing goes now
S> MLST
S> UTF8
S> NAT6
S> 211 End
C> CWD weather
C> CWD <e2.98.83>
   <the "<e2.98.83>" is a sequence of octets represented in
   Base 16, not a sequence of characters>
S> 250 Working directory changed
C> PORT 198,51,100,20,49,562
S> 200 Data connection aranged
C> RETR snow.txt
S> 150 Start transmission
   <server-DTP sending snow.txt over data connection to user-DTP>
S> 226 File sent

The following example illustrates 'ftp' URI with UCS characters, explicitly, the U+0125 LATIN SMALL LETTER H WITH CIURCUMFLEX in the <host> and U+1D120 MUSICAL SYMBOL G CLEF OTTAVA BASSA in one of <cws>s. The URI:

ftp://%C4%A5ost.example.com/music/%F0%9D%84%A0/clef.pdf
5. Security and Privacy Considerations

Generic security considerations for URIs are discussed in Section 7 of RFC 3986 [RFC3986]; for IRIs - in Section 8 of RFC 3987 [RFC3987].

Security considerations for FTP are addressed in RFC 2577 [RFC2577]. RFC 2228 [RFC2228] and RFC 4217 [RFC4217] provided several ways for securing FTP. However, the ‘ftp’ URI does not allow to denote whether any of these ways should be used. The ‘ftps’ URI scheme, which denotes the resource available via FTP secured as defined in RFC 4217, is known to be deployed; this document provisionally registers this scheme with IANA (see Section 6.2), but specifying it is out of the scope of this document.

Because of some security concerns RFC 3986 did deprecate the use of "user:pass" format of <userinfo>, as stated in Section 3.1; it only applies to ‘ftp’ URIs because of historical reasons. Obviously, those URIs which contain the password "in the clear" should be kept and transmitted securely (for example, using Transport Layer Security...
The "anonymous FTP" [RFC1635] has a number of security implications, too. When transmitting the email address as a password, if it is required by the server, there is a risk of email address harvesting by "middle-boxes" (man-in-the-middle attacks) and the ultimate server. As servers usually do not pay attention to such passwords, clients are encouraged to transmit email addresses which are either randomly-generated or non-existing, as doubled in Section 3.2.2.

Security considerations for usage of Internalized Domain Names are discussed in RFC 5890 [RFC5890].

6. Internalization Considerations

This document discusses internalization of ‘ftp’ URIs, as required by RFC 2277 [RFC2277].

6.1. UCS Characters in ‘ftp’ URIs

As allowed by Section 2.5 of RFC 3986, ‘ftp’ URIs MAY contain the characters from Universal Character Set (UCS) [UCS]. They are only allowed in <host> and <ftp-path> part; <userinfo> is not internalized.

In order to use such character in one of these parts, it SHALL first be encoded with UTF-8 [RFC3629]. The resulting sequence of octets SHALL be examined to conclude whether some octets match corresponding ASCII characters. If one does, and such character is allowed in a particular part of ‘ftp’ URI, it SHALL be presented in the URI directly; otherwise, the octet SHALL be represented percent-encoded.

6.2. ‘ftp’ IRIs

IRIs, described in RFC 3987 [RFC3987], may contain UCS characters "in the raw", unlike URIs, which only allow ones which are outside of ASCII range percent-encoded (see Section 4.1 above). Correspondingly, the syntax of ‘ftp’ IRIs will be different from URIs’ one.

The changes required in URI syntax to match valid URI is to change <host> of RFC 3986 to <ihost> of RFC 3987 in <host-port> and <unreserved> of RFC 3986 - to <iunreserved> of RFC 3987 in <segment-nsc>. The <userinfo> is not internalized.

6.2.1. Mapping ‘ftp’ IRIs to ‘ftp’ URIs

‘ftp’ IRIs are subject to the rules of Section 3.1 of RFC 3987 with...
6.3. Handling ‘ftp’ URIs with UCS Characters and ‘ftp’ IRIs

Handling of ftp’ URIs with UCS characters and ‘ftp’ IRIs is mostly the same as discussed in Section 3.2 of this document; however, a number of issues should be considered.

6.3.1. Internalized <host> Part in ‘ftp’ URIs and <ihost> Part in ‘ftp’ IRIs

The <host> part in ‘ftp’ URIs and <ihost> part in ‘ftp’ IRIs may contain internalized strings, with UCS characters being percent-encoded and displayed directly, respectively.

As Domain Name System (DNS) does not allow the UTF-8 encoded data in its interchange, limiting the allowed characters range to ASCII [ASCII], the usual procedure of UTF-8 transformation is insufficient here. Hence, in order to make up the valid domain name for lookup and further processing the following step SHALL be applied:

1. represent all percent-encoded characters directly, assuming they are in UTF-8 (if an URI);

2. apply the algorithm for IDN lookup defined in Section 5 of RFC 5891 [RFC5891].

The received sequence of A-labels SHALL also be used with FTP HOST command [I-D ietf-ftpext2-hosts], sent when establishing FTP connection per Section 3.2.1.

6.3.2. Internalized <ftp-path> Part in ‘ftp’ URIs and <iftp-path> in ‘ftp’ IRIs

The <ftp-path> and <iftp-path> parts allow to include UCS characters in FTP pathnames present in URIs and IRIs, respectively.

In order to successfully process an internalized FTP pathname, a client prior to its processing SHALL examine the server’s response to the FEAT command [RFC2389], issued upon authentication per Section 3.2. If one of the lines of the response is "UTF8", the server supports UTF-8 encoded pathnames [RFC2640]. Otherwise, if there is no such line, or the server does not support the FEAT mechanism, the contrary SHALL be assumed.

Should it be determined that server supports UTF-8 encoded pathnames, the internalized pathname parts SHALL be encoded with UTF-8 [RFC3629] and then transmitted as an arguments to the corresponding FTP
commands as UTF-8 octets stream.

6.4. Internalization of Actual Data Interchange

‘ftp’ RIs may refer to the file which contains the internalized data. When transmitting such file over data connection, it should be in Net-Unicode format [RFC5198]. In order to indicate this, the <typecode> equal to "u" [I-D ietf-ftpext2-typeu] SHALL be set in the ‘ftp’ URI or IRI.

7. IANA Considerations

7.1. The ‘ftp’ URI Scheme

IANA is asked to update the registration of the ‘ftp’ URI scheme in the appropriate registry [IANA-URIREG] with the reference to this document using the following template, per [RFC4395]:

URI scheme name: ftp

Status: Permanent

URI scheme syntax: see Section 3.1 of RFC xxxx

URI scheme semantics: see Section 3.2 of RFC xxxx

URI scheme encoding considerations: see Section 3.3 of RFC xxxx

Protocols that use the scheme: File Transfer Protocol (FTP) [RFC0959]

Security considerations: see Section 6 of RFC xxxx

Contact: IESG <iesg@ietf.org>

Author/Change controller: IETF <ietf@ietf.org>

References: see Section 8 of RFC xxxx

[RFC Editor: Please replace xxxx with assigned RFC number]

7.2. The ‘ftps’ URI Scheme

IANA is requested to provisionally register the ‘ftps’ URI scheme per RFC 4395 [RFC4395] with reference for this document. Specifying this scheme is out of the scope of this document; therefore the registration template is not supplied. As required by Section 4 of RFC 4395, the change controller for the registration is IETF
7.3. Maintaining ftp.uri.arpa Domain

As primarily requested by [MSG-REG] per RFC 3405 [RFC3405], IANA will continue maintaining the ftp.uri.arpa domain for use of DDDS with 'ftp' URIs (see Appendix A). Moreover, IANA is requested to change the existing substitution expression in the NAPTR record for this domain as described in Appendix A.

8. References

8.1. Normative References


ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.


8.2. Informative References


[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
August 1980.


Dynamic Delegation Discovery System (DDDS) and 'ftp' URIs

Dynamic Delegation Discovery System (DDDS) is an abstract algorithm for applying dynamically retrieved string transformation rules to an application-unique string. The comprehensive DDDS specification consists of 5 documents, which are defined in RFC 3401 [RFC3401].

RFC 3404 [RFC3404] specified a DDDS Application for resolving URIs using DDDS Algorithm defined in RFC 3402 [RFC3402]. A corresponding second-level domain has been delegated in the "arpa" zone [RFC3172] - "uri.arpa" [RFC3405] - for use with this Application. RFC 3404
specified that First Well Known Rule for the aforementioned DDDS Application is to append the URI scheme name to ".uri.arpa". According to the provisions of RFC 3405 [RFC3405], the 'ftp' URI scheme was previously approved for inclusion in this zone [MSG-REG] in order to allow its resolving using DDDS. Correspondingly, the following substitution expression was recorded in the NAPTR DNS resource record [RFC3403]:

```
!^ftp://([^:/?#]*)\.*$!\1!i
```

using the syntax defined in Section 3.2 of RFC 3402. However, taking the syntax specified in this document, IANA is asked to record the following new substitution expression in the NAPTR record for ftp.uri.arpa domain:

```
!^ftp://([^@/?#]@)?([^:/?#]*)\.*$!2!i
```

This substitution expression extracts the hostname from a given URI, skipping the <userinfo>, and forming the next Key.

Refer to RFC 3404 for detailed description of DDDS Application for resolving URIs and RFC 3402 for generic DDDS Algorithm.

Please note that while there is a possibility to resolve 'ftp' URIs using DDDS, not every given 'ftp' URI may be resolved using this technique. A specific "hint" is required in order to denote this; for instance, "the URI <ftp://exmaple.org/foo/bar.txt> refers to the very valuable information; it is mirrored at a number of servers which are to be discovered using DDDS".

### Appendix B. Previous Syntax Definitions

This appendix copies the definition of the syntax of 'ftp' URI from previous documents which specified it, which might be of some historical interest. Within this appendix, BNF refers to the convention described in Section 2 of RFC 822 [RFC0822].

#### B.1. RFC 1630

RFC 1630 [RFC1630] defined the syntax of 'ftp' URI with the following conventions:

- This is a BNF-like description of the URI syntax. at the level at which specific schemes are not considered.

- A vertical line "|" indicates alternatives, and [brackets] indicate optional parts. Spaces are represented by the word "space", and the vertical line character by "vline". Single letters stand for
as follows:

```plaintext
ftpaddress       ftp://login/path [ ftptype ]
login            [ user [ : password ] @ ] hostport
user             alphanumeric [ user ]
password         alphanumeric [ password ]
hostport         host [ : port ]
host             hostname | hostnumber
hostname         ialpha [ . hostname ]
hostnumber       digits . digits . digits . digits
path             void | segment [ / path ]
segment          xalphas
void

ftptype          A formcode | E formcode | I | L digits
formcode         N | T | C

alphanumeric2    alphanumeric2 [ user ]
alpha            a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
digit            0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
ialpha           alpha [ xalphas ]
xalphas          xalpha [ xalphas ]
xalpha           alpha [ digit | safe | extra | escape
safe            $ | - | _ | @ | . | & | + | -
extra           ! | * | " | ' | ( | ) | ,
escape           % hex hex
hex              digit | a | b | c | d | e | f | A | B | C | D | E | F
digits           digit [ digits ]
```

B.2. RFC 1738

RFC 1738, which was the first Standard Track specification for many URI schemes, defined the syntax of ‘ftp’ URIs with the following conventions:

This is a BNF-like description of the Uniform Resource Locator...
syntax, using the conventions of RFC822, except that "|" is used to
designate alternatives, and brackets [] are used around optional or
repeated elements. Briefly, literals are quoted with "", optional
elements are enclosed in [brackets], and elements may be preceded
with <n>* to designate n or more repetitions of the following
element; n defaults to 0.

as follows:

ftpurl = "ftp://" login [ "/" fpath [ ";type=" ftptype ]]

login = [ user [ ":" password ] "@" ] hostport
hostport = host [ ":" port ]
host = hostname | hostnumber
hostname = [ domainlabel "." ] toplabel
domainlabel = alphadigit | alphadigit *[ alphadigit | "-" ]
alphadigit
toplabel = alpha | alpha *[ alphadigit | "-" ] alphadigit
hostnumber = digits "." digits "." digits
port = digits
user = *[ uchar | ":" | ":?" | ":@" | ":&" ]
password = *[ uchar | ":" | ":?" | ":@" | ":&" ]
urlpath = *xchar

fpath = fsegment *[ "/" fsegment ]
fssegment = *[ uchar | ":?" | ":@" | ":&" ]
ftptype = "A" | "I" | "D" | "a" | "i" | "d"
alphadigit = alpha | digit
alpha = lowalpha | hialpha
lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
          | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
          | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
          | "y" | "z"
hialpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
          | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
          | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
          | "Y" | "Z"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
          | "8" | "9"
digits = 1*digit
uchar = unreserved | escape
unreserved = alpha | digit | safe | extra
safe = "&" | "*" | "-" | ":" | ";" | ":?" | ":@" | ":&"
extra = "!*" | "!*" | ":!*" | ":!*" | ":!" | "!" | ":!"
escape = "%" hex hex
hex = digit | "A" | "B" | "C" | "D" | "E" | "F" |
Appendix C. List of Changes since RFC 1738

The first specification of the ‘ftp’ URI is RFC 1738. This appendix lists main changes since that document.

Updated syntax specification to use ABNF.
Specification changed to suit RFC 3986.
Changes made to accommodate HOST command [I-D ietf-ftpext2-hosts].
Given more detailed description of <user-pass> semantics.
Clarified the <ftp-path> syntax.
Given detailed algorithm of handling <ftp-path>.
Clarified client’s handling null <cwd>s in <ftp-path>.
Added internalization considerations.
Clarified encoding considerations.
Clarified security considerations.
Added provisions regarding DDDS.
Various editorial changes/corrections.

Appendix D. Acknowledgments

The authors of RFC 1630 and RFC 1738, who worked on the initial ‘ftp’ URI scheme definition, included Tim Berners-Lee, Larry Masinter and Mark McCahill. Previous attempts to specify this URI scheme were undertaken by James Casey and Paul Hoffman.

Considerable input to this document was provided by (in alphabetical order) Frank Ellermann, John Klensin, Gordon Spoelhof, and Daniel Stenberg.

Author’s Addresses

Mykyta Yevstifeyev
8 Kuzovkov St., Apt. 25
Kotovsk
Ukraine

EMail: evnikita2@gmail.com