Network Working Group
Request for Comments: 2224
Category: Informational

B. Callaghan
Sun Microsystems, Inc.

October 1997

NFS URL Scheme

Status of this Memo

Copyright Notice

Abstract

A new URL scheme, 'nfs' is defined. It is used to refer to files and directories on NFS servers using the general URL syntax defined in RFC 1738, "Uniform Resource Locators (URL)".

This scheme uses the public filehandle and multi-component lookup [RFC2054, RFC2055] to access server data with a minimum of protocol overhead.

The NFS protocol provides access to shared filesystems across networks. It is designed to be machine, operating system, network architecture, and transport protocol independent. The protocol currently exists in two versions: version 2 [RFC1094] and version 3 [RFC1813], both built on ONC RPC [RFC1831] at its associated eXternal Data Representation (XDR) [RFC1832] and Binding Protocol [RFC1833].

Table of Contents

1. URL Syntax

An NFS URL is based on the Common Internet Scheme Syntax described in section 3.1 of RFC 1738. It has the general form:

nfs://<host>:<port><url-path>

The ":<port>" part is optional. If omitted then port 2049 is assumed. The <url-path> is also optional.

The <url-path> is a hierarchical directory path of the form /<directory>/<directory>/<directory>/.../<name>. The <url-path> must consist only of characters within the US-ASCII character set. Within a <directory> or <name> component the character "/" is reserved and must be encoded as described in Section 2.2 of RFC 1738. If <url-path> is omitted or consists solely of "/", it must default to the path ".".

2. URL Evaluation

A client must evaluate an NFS URL by a method known as WebNFS [RFC2054, RFC2055]. This method provides easy passage through firewalls and proxy servers, as well as using a minimum number of messages. The WebNFS method is defined for NFS versions 2 and 3. It assumes that the server registers on TCP or UDP port 2049 and supports the public filehandle and multi-component lookup semantics as described in the following sections.

3. Server Connection

The client must first attempt to create a TCP connection to <host> using the <port> specified. If :<port> is omitted, then port 2049 will be used. If the server refuses the TCP connection, then the client will use UDP.

4. NFS Version

The client must first attempt to use NFS version 3. If the server returns an RPC PROG_MISMATCH error then the client must assume that NFS version 3 is not supported, and retry the operation with an NFS version 2 public filehandle.

5. Public Filehandle

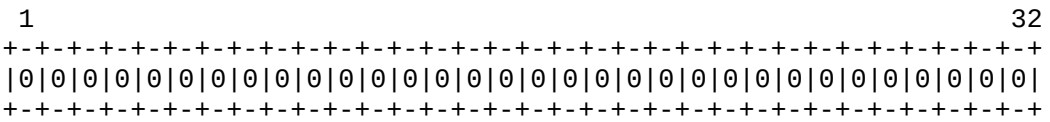
NFS filehandles are normally created by the server and used to identify uniquely a particular file or directory on the server. The client does not normally create filehandles or have any knowledge of the contents of a filehandle.

The public filehandle is an an exception. It is an NFS filehandle with a reserved value and special semantics that allow an initial filehandle to be obtained. A WebNFS client uses the public filehandle as an initial filehandle rather than using the MOUNT protocol. Since NFS version 2 and version 3 have different filehandle formats, the public filehandle is defined differently for each.

The public filehandle is a zero filehandle. For NFS version 2 this is a filehandle with 32 zero octets. A version 3 public filehandle has zero length.

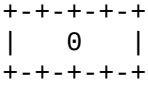
5.1 NFS Version 2 Public Filehandle

A version 2 filehandle is defined in RFC 1094 as an opaque value occupying 32 octets. A version 2 public filehandle has a zero in each octet, i.e. all zeros.



5.2 NFS Version 3 Public Filehandle

A version 3 filehandle is defined in RFC 1813 as a variable length opaque value occupying up to 64 octets. The length of the filehandle is indicated by an integer value contained in a 4 octet value which describes the number of valid octets that follow. A version 3 public filehandle has a length of zero.



6. Multi-component Lookup

Normally the NFS LOOKUP request (version 2 or 3) takes a directory filehandle along with the name of a directory member, and returns the filehandle of the directory member. If a client needs to evaluate a pathname that contains a sequence of components, then beginning with

the directory filehandle of the first component it must issue a series of LOOKUP requests one component at a time. For instance, evaluation of the path "a/b/c" will generate separate LOOKUP requests for each component of the pathname "a", "b", and "c".

A LOOKUP request that uses the public filehandle can provide a pathname containing multiple components. The server is expected to evaluate the entire pathname and return a filehandle for the final component.

For example, rather than evaluate the path "a/b/c" as:

```
     LOOKUP  FH=0x0  "a"  --->
                          <---  FH=0x1
     LOOKUP  FH=0x1  "b"  --->
                          <---  FH=0x2
     LOOKUP  FH=0x2  "c"  --->
                          <---  FH=0x3
```

Relative to the public filehandle these three LOOKUP requests can be replaced by a single multi-component lookup:

```
     LOOKUP  FH=0x0  "a/b/c"  --->
                              <---  FH=0x3
```

Multi-component lookup is supported only for LOOKUP requests relative to the public filehandle.

The <url-path> of the NFS URL must be evaluated as a multi-component lookup. This implies that the path components are delimited by slashes and the characters that make up the path must be in the printable US-ASCII character set. Characters not in the "unreserved" set (see BNF description below) may be included in pathname components but must be escaped.

If the <url-path> is empty or consists solely of "/", the client must send a multi-component lookup for the pathname ".".

6.1 Absolute vs. Relative Pathname

A multi-component pathname that begins with a slash character is considered "absolute" and will be evaluated relative to the server's root. A pathname that does not begin with a slash is "relative" and will be evaluated relative to the directory with which the public filehandle is associated.

Note that the initial "/" that introduces the <url-path> of an NFS URL must not be passed to the server for multi-component lookup since the pathname is to be evaluated relative to the public filehandle directory. For example, if the public filehandle is associated with the server's directory "/a/b/c" then the URL:

nfs://server/d/e/f

will be evaluated with a multi-component lookup of the path "d/e/f" relative to the server's directory "/a/b/c" while the URL:

nfs://server//a/b/c/d/e/f

will locate the same file with an absolute multi-component lookup of the path "/a/b/c/d/e/f" relative to the server's filesystem root. Notice that a double slash is required at the beginning of the path.

Not all WebNFS servers can support arbitrary use of absolute paths. Clearly, the server must not return a filehandle if the path identifies a file or directory that is not exported by the server. In addition, some servers will not return a filehandle if the path names a file or directory in an exported filesystem different from the one that is associated with the public filehandle.

6.2 Symbolic Links

The NFS protocol supports symbolic links, which are the filesystem equivalent of a relative URL. If a WebNFS client retrieves a filehandle for a symbolic link (as indicated by the file type attribute) then it should send a READLINK request to the server to retrieve the path comprising the symbolic link.

This path should then be combined with the URL which referenced the symbolic link according to the rules described in RFC 1808. If the relative URL in the symbolic link text is to be resolved successfully then it must contain only ASCII characters and conform to the syntax described in RFC 1808. Note that this allows a symbolic link to contain an entire URL and it may specify a scheme that is not necessarily an NFS URL.

A symbolic link path that begins with a slash will be evaluated as an absolute path relative to the directory associated with the public filehandle which may not be the same as the server's system root directory. If symbolic links with absolute paths are to be evaluated correctly on both client and server then the public filehandle must be associated with the system root directory.

For example, if the symbolic link is named by the URL

```
nfs://server/a/b
```

then the following examples show how a new URL can be formed from the symbolic link text:

```
c                    = nfs://server/a/c

c/d                  = nfs://server/a/c/d

../c                 = nfs://server/c

/c/d                 = nfs://server/c/d

nfs://server2/a/b    = nfs://server2/a/b
```

7. Mount Protocol

The NFS URL may have limited use for naming files on servers that do not support the public filehandle and multi-component lookup.

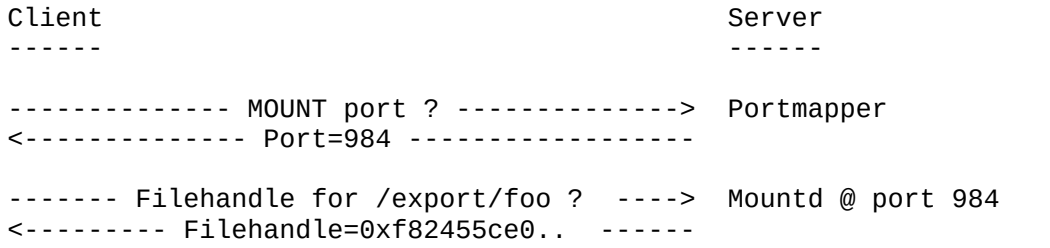
If the server returns an NFS3ERR_STALE, NFS3ERR_INVAL, or NFS3ERR_BADHANDLE error in response to the client's use of a public filehandle, then the client should attempt to resolve the <url-path> to a filehandle using the MOUNT protocol.

Version 1 of the MOUNT protocol is described in Appendix A of RFC 1094 and version 3 in Appendix I of RFC 1813. Version 2 of the MOUNT protocol is identical to version 1 except for the addition of a procedure MOUNTPROC_PATHCONF which returns POSIX pathconf information from the server.

Note that the pathname sent to the server in the MOUNTPROC_MNT request is assumed to be a server native path, rather than a slash-separated path described by RFC 1738. Hence, the MOUNT protocol can reasonably be expected to map a <url-path> to a filehandle only on servers that support slash-separated ASCII native paths. In general, servers that do not support WebNFS access or slash-separated ASCII native paths should not advertise NFS URLs.

At this point the client must already have some indication as to which version of the NFS protocol is supported on the server. Since the filehandle format differs between NFS versions 2 and 3, the client must select the appropriate version of the MOUNT protocol. MOUNT versions 1 and 2 return only NFS version 2 filehandles, whereas MOUNT version 3 returns NFS version 3 filehandles.

Unlike the NFS service, the MOUNT service is not registered on a well-known port. The client must use the PORTMAP service to locate the server's MOUNT port before it can transmit a MOUNTPROC_MNT request to retrieve the filehandle corresponding to the requested path.



NFS servers commonly use a client's successful MOUNTPROC_MNT request request as an indication that the client has "mounted" the filesystem and may maintain this information in a file that lists the filesystems that clients currently have mounted. This information is removed from the file when the client transmits an MOUNTPROC_UMNT request. Upon receiving a successful reply to a MOUNTPROC_MNT request, a WebNFS client should send a MOUNTPROC_UMNT request to prevent an accumulation of "mounted" records on the server.

8.0 Bibliography

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

[RFC1808] Fielding, R., "Relative Uniform Resource Locators," RFC 1808, June 1995.

[RFC1831] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2," RFC 1831, August 1995.

[RFC1832] Srinivasan, R., "XDR: External Data Representation Standard," RFC 1832, August 1995.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2," RFC 1833, August 1995.

[RFC1094] Sun Microsystems, Inc., "Network Filesystem Specification," RFC 1094, March 1989.

[RFC1813] Callaghan, B., Pawlowski, B. and P. Staubach, "NFS Version 3 Protocol Specification," RFC 1813, June 1995.

[RFC2054] Callaghan, B., "WebNFS Client Specification," RFC 2054, October 1996.

[RFC2055] Callaghan, B., "WebNFS Server Specification," RFC 2055, October 1996.

[Sandberg] Sandberg, R., D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, "Design and Implementation of the Sun Network Filesystem," USENIX Conference Proceedings, USENIX Association, Berkeley, CA, Summer 1985. The basic paper describing the SunOS implementation of the NFS version 2 protocol, and discusses the goals, protocol specification and trade-offs.

[X/OpenNFS] X/Open Company, Ltd., X/Open CAE Specification: Protocols for X/Open Internetworking: XNFS, X/Open Company, Ltd., Apex Plaza, Forbury Road, Reading Berkshire, RG1 1AX, United Kingdom, 1991. This is an indispensable reference for the NFS and accompanying protocols, including the Lock Manager and the Portmapper.

[X/OpenPCNFS] X/Open Company, Ltd., X/Open CAE Specification: Protocols for X/Open Internetworking: (PC)NFS, Developer's Specification, X/Open Company, Ltd., Apex Plaza, Forbury Road, Reading Berkshire, RG1 1AX, United Kingdom, 1991. This is an indispensable reference for NFS protocol and accompanying protocols, including the Lock Manager and the Portmapper.

9. Security Considerations

Since the WebNFS server features are based on NFS protocol versions 2 and 3, the RPC based security considerations described in RFC 1094, RFC 1831, and RFC 1832 apply here also.

Server implementors should be careful when implementing multi-component lookup that the client cannot obtain unauthorized access to files or directories. For example: a path that includes multiple occurrences of "../" may locate a filesystem outside of the exported filesystem associated with the public filehandle.

Clients and servers may separately negotiate secure connection schemes for authentication, data integrity, and privacy.

10. BNF for NFS URL Scheme

The syntax of the NFS URL is a subset of the Generic URL syntax described in RFC 1738. An NFS URL does not include user or password components, nor does it recognize "?" query or "#" fragment components.

```
   nfsURL        = "nfs" ":" relativeURL
   relativeURL   = net_path | abs_path | rel_path
   net_path      = "//" hostport [ abs_path ]
   abs_path      = "/"  rel_path
   rel_path      = [ path_segments ]

   hostport      = host [ ":" port ]
   host          = hostname | hostnumber
   hostname      = *( domainlabel "." ) toplabel
   domainlabel   = alphanum | alphanum *( alphanum | "-" ) alphanum
   toplabel      = alpha | alpha *( alphanum | "-" ) alphanum
   hostnumber    = 1*digit "." 1*digit "." 1*digit "." 1*digit
   port          = *digit

   url-path      = [ "/" ] path_segments
   path_segments = segment *( "/" segment )
   segment       = *pchar
   pchar         = unreserved | escaped | ":" | "@" | "&" | "=" | "+"

   reserved      = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+"
   unreserved    = alpha | digit | mark
   mark          = "$" | "-" | "_" | "." | "!" | "~" |
                   "*" | "'" | "(" | ")" | ","

   escaped       = "%" hex hex
   hex           = digit | "A" | "B" | "C" | "D" | "E" | "F" |
                           "a" | "b" | "c" | "d" | "e" | "f"

   alphanum      = alpha | digit
   alpha         = lowalpha | upalpha

   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"
   upalpha  = "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"
```

11. Acknowledgements

This specification was extensively reviewed by the NFS group at SunSoft and brainstormed by Michael Eisler.

12. Author's Address

Address comments related to this RFC to:

brent@eng.sun.com

Brent Callaghan
Sun Microsystems, Inc.
Mailstop Mpk17-201,
901 San Antonio Road,
Palo Alto, California 94303

Phone: 1-415-786-5067
EMail: brent.callaghan@eng.sun.com
Fax: 1-415-786-5896

13. Full Copyright Statement