INTERNET-DRAFT
Expires: January 2005

David Noveck
Network Appliance, Inc.

July 2004

# Migration Issues for NFSv4
# draft-noveck-nfsv4-migration-issues-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, or will be disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

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/ietf/1id-abstracts.txt The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

RFC3530 described an fs_locations attribute which can be used to allow an fs to migrate between servers without disrupting client access and to refer a client to another server providing access to a specified file system. Initial implementation work for this feature has exposed a number of areas where RFC3530's handling of the issues leaves something to be desired. This document makes a number of suggestions to remedy these issues when the NFSv4 spec next undergoes a significant revision, most likely in connection with implementing a new minor revision, such as NFSv4.1.

Table Of Contents



## 1. Introduction

Ongoing design work has exposed a number of weaknesses in the discussion of migration within RFC 3530. While there does not appear any necessity to change any message formats or add operations, a number of migration-related issues should be addressed when the protocol is updated for v4.1. The purpose of this document is to clearly lay out what needs to be done, so that any possible updates can be discussed as part of the process of formulating a spec for v4.1. Some of the items discussed below might also be appropriate in the context of an update of the NFSv4 spec in connection with going to a Draft Standard status.

## 2. Attributes Returned by GETATTR and READDIR

While the RFC3530 allows the server to return attributes in addition to fs_locations, when GETATTR is used with a current filehandle within an absent filesystem, not much guidance is given to help clarify what is appropriate. In particular, there are a number of attributes which most server implementations should find relatively easy to supply which would be of value to clients,

particularly in those cases in which NFS4ERR_MOVED is returned when first crossing into an absent file system that the client has not referenced.

The NFSv4 spec should encourage servers to return the attributes fsid and mounted_on_fileid for absent file systems. The specific reasons will be discussed below.

On the other hand, a number of attributes pose difficulties when returned for an absent filesystem. While not prohibiting the server from returning these, the NFSv4 spec should explain the issues which may result in problems, since these are not always obvious. Handling of specific attributes is discussed below.

## 2.1. fsid

The fsid attribute allows clients to recognize when fs boundaries have been crossed. This applies also when one crosses into an absent filesystem. While returning fsid is not absolutely required, since fs boundaries are also reflected, in this case, by means of the fs_root field of the fs_locations attribute, returning fsid is helpful and servers should have no difficulty in providing it.

To avoid misunderstanding, any new NFSv4 RPC should note that the fsid provided in this case is solely so that the fs boundaries can be properly noted and that the fsid returned will not necessarily be valid after resolution of the migration event. The logic of fsid handling for NFSv4 is that fsid's are only unique within a per-server context. This would seem to be a strong indication that they need not be persistent when file systems are moved from server to server, although RFC 3530 does not specifically address the matter.

## 2.2. mounted_on_fileid

The mounted_on_fileid attribute is of particular importance to many clients, in that they need this information to form a proper response to a readdir() call. When a readdir() call is done within UNIX, the d_ino field of each of the entries needs to have a unique value normally derived from the NFSv4 fileid attribute. It is in the case in which a file system boundary is crossed that using the fileid attribute for this purpose, particularly when crossing into an absent fs, will pose problems. Note first that the fileid attribute, since it is within a new fs and thus a new fileid space, will not be unique within the directory. Also, since the fs, at its new location, may arrange things differently, the fileid decided on at the directing server may be overridden at the target

server, making it of little value. Neither of these problems arise in the case of mounted_on_fileid since that fileid is in the context of the mounted-on fs and unique within it.

### 2.3. fileid

For reasons explained above under mounted_on_fileid, it would be difficult for the referring server to provide a fileid value that is of any use to the client. Given this, it seems much better for the server never to return fileid values for files on an absent fs.

### 2.4. filehandle

Returning file handles for files in the absent fs, whether by use of GETFH (discussed below) or by using the filehandle attribute with GETATTR or READDIR poses problems for the client as the server to which it is referred is likely not to assign the same filehandle value to the object in question. Even though it is possible that volatile filehandles may allow a change, the referring server should not prejudge the issue of filehandle volatility for the server which actually has the fs. By not providing the file handle, the referring server allows the target server freedom to choose the file handle value without constraint.

## 3. Discussion of Error Codes

There are a number of cases in which RFC 3530 is either unclear or simply incorrect about the situations in which NFS4ERR_MOVED is to be returned. Discussion of these issues has exposed the following problems, which should be addressed to provide greater clarity and correctness:

### 3.1. Issue of when to check current filehandle

In providing the definition of NFS4ERR_MOVED, RFC 3530 refers to the "filesystem which contains the current filehandle object" being moved to another server. This has led to some confusion when considering the case of operations which change the current filehandle and potentially the current file system. For example, a LOOKUP which causes a transition to an absent file system might be supposed to result in this error. This should be clarified to make it explicit that only the current filehandle at the start of the operation can result in NFS4ERR_MOVED.

### 3.2. Issue of GETFH

While RFC 3530 does not make any exception for GETFH when the current filehandle is within an absent filesystem, the fact that

GETFH is such a passive, purely interrogative operation, may lead readers to wrongly suppose that an NFSERR_MOVED error will not arise in this situation. Any new NFSv4 RFC should explicitly state that GETFH will return this error if the current filehandle is within an absent filesystem.

### 3.3. Inconsistent handling of PUTFH

While RFC 3530 states (in section 6.2) "The NFS4ERR_MOVED error is returned for all operations except PUTFH and GETATTR." Despite this, RFC 3530 lists NFS4ERR_MOVED as an error that can be returned by PUTFH. Any new NFSv4 RFC should delete this as a possible error for PUTFH.

### 3.4. Inconsistent handling of GETATTR

While, as noted above, RFC 3530 indicates that NFS4ERR_MOVED is not returned for a GETATTR operation, NFS4ERR_MOVED is listed as an error that can be returned by GETATTR. It seems reasonable to allow NFS4ERR_MOVED to be returned by GETATTR's that do not interrogate the fs_locations attribute while maintaining the exception which allows GETATTR to be used to get fs_locations information by establishing the rules that GETATTR's which interrogate fs_locations (with or without additional attributes) will not return NFS4ERR_MOVED.

## 4. Issues of Incomplete Attribute Sets

Migration or referral events naturally create situations in which all of the attributes normally supported on a server are not obtainable. RFC3530 is in places ambivalent and/or apparently self-contradictory on such issues. Any new NFSv4 RFC should take a clear position on these issues (and it should not impose undue difficulties on support for migration).

The first problem concerns the statement in the third paragraph of section 6.2: "If the client requests more attributes than just fs_locations, the server may return fs_locations only. This is to be expected since the server has migrated the filesystem and may not have a method of obtaining additional attribute data."

While the above seems quite reasonable, it is seemingly contradicted by the following text from section 14.2.7 the second paragraph of the DESCRIPTION for GETATTR: "The server must return a value for each attribute that the client requests if the attribute is supported by the server. If the server does not support an attribute or cannot approximate a useful value then it must not return the attribute value and must not set the attribute bit

in the result bitmap. The server must return an error if it supports an attribute but cannot obtain its value. In that case no attribute values will be returned."

While the above is a useful restriction in that it allows clients to simplify their attribute interpretation code since it allows them to assume that all of the attributes they request are present often making it possible to get successive attributes at fixed offsets within the data stream, it seems to contradict what is said in section 6.2, where it is clearly anticipated, at least when fs_locations is requested, that fewer (often many fewer) attributes will be available than are requested. It could be argued that you could harmonize these two by being creative with the interpretation of the phrase "if the attribute is supported by the server". One could argue that many attributes are not supported by the server for an absent fs even though the text by talking about attributes "supported by a server" seems to indicate that this is not allowed to be different for different fs's (which is troublesome in itself as one server might have filesystems that do support and don't support acl's for example).

Note however that the following paragraph in the description says, "All servers must support the mandatory attributes as specified in the section 'File Attributes'". That's reasonable enough in general, but for an absent fs it is not reasonable and so section 14.2.7 and section 6.2 are contradictory. Any new NFSv4 RFC should remove the contradiction, while allowing servers to use the approach outlined in section 6.2. It should also make sure that it is clear that the server may choose to return other requested attributes (e.g. fsid and mounted_on_fileid) rather than fs_locations alone.

A related issue concerns attributes in a READDIR. RFC 3530 already allows partial attribute return when rdattr_error is requested but indicates that if it is not requested, errors must be returned if not all requested attributes can be obtained. When READDIR is done on a directory which contains mountpoints for absent fs's (either those that were once present and then migrated or simple referrals), this would seem to indicate that NFS4ERR_MOVED must be returned if the directory is in absent filesystem or any of the directory entries is the root of absent fs. This seems unduly restrictive, but if that is the correct interpretation, it should be made clear that the exception indicated in section 6.2 does not apply in the READIR case, to avoid possible confusion.

## 5. Referral Issues

RFC 3530 defines a migration feature which allows the server to direct clients to another server for the purpose of accessing a given file system. While that document explains the feature in terms of a client accessing a given file system and then finding that it has moved, an important limiting case is that in which the clients are redirected as part of their first attempt to access a given file system.

Such redirection is often described as a referral event and implementing such a form of migration has many important consequences, which are not well explained by the presentation within RFC 3530, which often assumes that the client is informed of the migration event after one or more accesses within the file system for which a migration event occurs.

### 5.1. Referral Situations

When a particular client is directed to a new location upon first referencing a file system, the result is best seen, from the client's point of view as a referral, rather than a migration event, since the client contains no information derived from the file system before the migration occurred.

Note that the above only refers to a particular client's point of view. A given file system may be accessed by some clients and thus, when a migration occurs, those clients will see an ordinary migration event while other clients see a referral when they first attempt to access the subject filesystem.

In the case in which none of the clients has referenced the subject file system at the time of migration, we have a pure referral situation, in that all that clients will ever see is the referral for an absent file system. Given that clients can use such referrals to find the current location of file systems, servers can usefully provide such referrals when the filesystem in question never actually resided on the server. Such referrals may allow implementation of some forms of multi-server namespace, although NFSv4 support of a global namespace would require considerable additional work. Nevertheless, an arrangement where the client addresses file systems in terms of the name of the filesystem on a server providing referrals may be valuable, because it allows the clients to isolate themselves from server configuration changes which move file systems from server to server.

## 5.2. Referral Interactions (LOOKUP)

The details of how referrals proceed are implicit in the specification of migration in RFC 3530. However, because the details of handling of this case are so different from those in the cases discussed therein, examples tailored to the referral situation are needed to clarify matters and allow correct and consistent implementations.

Let us suppose that the following COMPOUND is issued in an environment in which /src/linux/2.7/latest is absent from the target server. This may be for a number of reasons. It may be the case that the file system has moved, or, it may be the case that the target server is functioning mainly or solely to refer clients to the server on which the file system is located.

- PUTROOTFH
- LOOKUP "src"
- LOOKUP "linux"
- LOOKUP "2.7"
- LOOKUP "latest"
- GETFH
- GETATTR fsid,fileid,size,ctime

Under the given circumstances, the following will be the result.

- PUTROOTFH --> NFS_OK

  Current fh is root of pseudo-fs.

- LOOKUP "src" --> NFS_OK

  Current fh is for /src and is within pseudo-fs.

- LOOKUP "linux" --> NFS_OK

  Current fh is for /src/linux and is within pseudo-fs.

- LOOKUP "2.7" --> NFS_OK

  Current fh is for /src/linux/2.7 and is within pseudo-fs.

- LOOKUP "latest" --> NFS_OK

  Current fh is for /src/linux/2.7/latest and is within a new, absent fs, but ...

  The client will never see the value of that fh.

- GETFH --> NFS4ERR_MOVED

  Fails because current fh is in an absent fs at the start of the operation and the spec makes no exception for GETFH.

- GETATTR fsid,fileid,size,ctime

  Not executed because the failure of the GETFH stops processing of the COMPOUND.

Given the failure of the GETFH, the client has the job of determining the root of the absent file system and where to find that file system, i.e. the server and path relative to that server's root fh. Note here that in this example, the client did not obtain filehandles and attribute information (e.g. fsid) for the intermediate directories, so that he would not be sure where the absent file system starts. It could be the case, for example, that /src/linux/2.7 is the root of the moved filesystem and that the reason that the lookup of "latest" succeeded is that the filesystem was not absent on that op but was moved between the last LOOKUP and the GETFH (since COMPOUND is not atomic). Even if we had the fsid's for all of the intermediate directories, we could have no way of knowing that /src/linux/2.7/latest was the root of a new fs, since we don't yet have its fsid.

In order to get the necessary information, let us re-issue the chain of lookup's with GETFH's and GETATTR's to at least get fsid's and fs_locations values.

- PUTROOTFH --> NFS_OK

  Current fh is root of pseudo-fs.

- GETATTR(fsid) --> NFS_OK

  Just for completeness. Normally, clients will know the fsid of the pseudo-fs as soon as they establish communication with a server.

- LOOKUP "src" --> NFS_OK

- GETATTR(fsid, fs_locations) --> NFS_OK

  Get current fsid to see where fs boundaries are. The fsid will be that for the pseudo-fs in this example, so no boundary. Get fs_locations just in case "src" is part of fs that moved.

- GETFH --> NFS_OK

  Current fh is for /src and is within pseudo-fs.

- LOOKUP "linux" --> NFS_OK

  Current fh is for /src/linux and is within pseudo-fs.

- GETATTR(fsid, fs_locations) --> NFS_OK

  Get current fsid to see where fs boundaries are. The fsid will be that for the pseudo-fs in this example, so no boundary. Get fs_locations just in case "linux" is part of the fs that moved.

- GETFH --> NFS_OK

  Current fh is for /src/linux and is within pseudo-fs.

- LOOKUP "2.7" --> NFS_OK

  Current fh is for /src/linux/2.7 and is within pseudo-fs.

- GETATTR(fsid, fs_locations) --> NFS_OK

  Get current fsid to see where fs boundaries are. The fsid will be that for the pseudo-fs in this example, so no boundary. Get fs_locations just in case "2.7" is part of fs that moved.

- GETFH --> NFS_OK

  Current fh is for /src/linux/2.7 and is within pseudo-fs.

- LOOKUP "latest" --> NFS_OK

  Current fh is for /src/linux/2.7/latest and is within a new, absent fs, but ...

  The client will never see the value of that fh

- GETATTR(fsid, fs_locations) --> NFS_OK

  We are getting the fsid to know where the fs boundaries are. The server may oblige us by giving us an fsid value different from that of the pseudo-fs. However, RFC 3530 does not oblige him to give us anything but fs_locations, so this may not be available. Note that if we did have the fsid, it would not necessarily be preserved at the new location. That fsid might be different and in fact the fsid we have for this fs might be the fsid of a different fs on that new server.

  In this particular case, we are pretty sure anyway that what has moved is /src/linux/2.7/latest rather than /src/linux/2.7 since we have the fsid of the latter and it is that of the pseudo-fs, which presumably cannot move. However, in other examples, we might not have this kind of information to rely on (e.g. /src/linux/2.7 might be a non-pseudo filesystem separate from /src/linux/2.7/latest), so we need to have another reliable source information on the boundary of the fs which is moved.

  The fs_locations attribute indicates the server and server-relative path of the fs's new location but it also gives us the necessary information about the fs boundaries via the fs_root field. In this case the fs_root field is /src/lnux/2.7/latest, telling us where the moved fs starts.

- GETFH --> NFS4ERR_MOVED

  Fails because current fh is in an absent fs at the start of the operation and the spec makes no exception for GETFH. Note that this has the happy consequence that we don't have to worry about the volatility or lack thereof of the fh. If the root of the fs on the new location is a persistent fh, then we can assume that this fh, which we never saw is a persistent fh, which, if we could see it, would exactly match the new fh. At least, there is no evidence to disprove that. On the other hand, if we find a volatile root at the new location, then the filehandle which we never saw must have been volatile or at least nobody can prove otherwise.

Given the above, the client knows where the root of the absent file system is, either by noting where the change of fsid occurred, or, if that is not provided, by means of the fs_root field of the fs_locations attribute. The fs_locations attribute also gives the client the actual location of the absent file system, so that the referral can proceed. Generally, the server will give the client the bare minimum of information about the absent file system so

that there will be very little scope for problems of conflict between information sent by the referring server and information of the file system's home. No filehandles and very few attributes are present on the referring server and the client can treat those it receives as basically transient information with the function of enabling the referral.

### 5.3. Referral Interactions (READDIR)

Another context in which a client may encounter referrals is when it does a READDIR on directory in which some of the sub-directories are the roots of absent file systems. In this case, NFS4ERR_MOVED is not an appropriate return because the directory in question is available and only the entries, whose attributes are being interrogated, are for absent file systems.

The appropriate approach for the server in this case is to provide the attributes which it can provide and not provide those that require access to the actual file system, while allowing the client to deduce that an fs boundary is present and that the file system is absent. Fortunately, this can be done fairly easily, as long as the client and server take proper care.

The basic strategy is to return only the attributes that can validly be provided by the referring server. Others are simply not provided. The spec allows this to be done if the client requests the rdattr_error as part of the READDIR, so, if the client does not request this attribute routinely, it must do so when re-issuing a READDIR which gets an NFS4ERR_MOVED error. Without a request for rdattr_error, NFS4ERR_MOVED could mean that the directory being read is within an absent file system or that one or more of the entries in the directory is the root of an absent file system. There is simply no way of determining which unless rdattr_error is requested.

Assuming the rdattr_error is requested, the server should, for the roots of absent file systems, return the attributes listed below when they are requested and no others. Returning other attributes may be possible in particular cases, but generally speaking, they are not necessary for clients to function and since clients will have to be prepared to get the necessary information from the actual root of the fs on the other server, servers are best advised to simply return this small set.

- fs_locations

  Location of the file system for the client's use when he needs to do the nested mount or to get attribute information about

the root of the fs.

- fsid

  Needs to be a value different from the fsid of the containing directory, in order to indicate to the client that a file system boundary is present.

  Note that the client should not expect that the actual fs, when located, will have the same fsid. Fsid's are unique only within the set of file systems exported by a single server so it might be possible to maintain the fsid on a different server. In any case, a client will be capable of using file systems on multiple servers and will therefore, if it needs to present unique identifiers for file systems to present to applications, needs to create them and not assume that using fsid's will provide the requisite uniqueness. So a change of mapping from one fsid-server pair to a given id, to another fsid-server pair as part of a referral or migration should not pose difficulties.

- mounted_on_fileid

  The mounted_on_fileid attribute is of particular importance to many clients, in that they need this information to form a proper response to a readdir() call. When a readdir() call is done within UNIX, the d_ino field of each of the entries needs to have a unique value normally derived from the NFSv4 fileid attribute. It is in the case in which a file system boundary is crossed that using the fileid attribute, particularly when crossing into an absent fs, that use of the fileid attribute for this purpose will pose problems. Note first that the fileid attribute, since it is within a new fs and thus a new fileid space, will not be unique within the directory. Also, since the fs, at its new location, may arrange things differently, the fileid decided on at the directing server may be overridden at the target server, making it of little value. Neither of these problems arise in the case of mounted_on_fileid since that fileid is in the context of the mounted-on fs and unique within it.

- rdattr_error

  The value should always be NFS4ERR_MOVED for entries that correspond to the root of absent file systems

## 5.4. Editorial Changes Related to Referrals

Given the above framework for implementing referrals, within the basic migration framework described in RFC 3530, we need to consider how future NFSv4 RFC's should be modified, relative to RFC 3530, to address referrals.

The most important change is to include an explanation of how referrals fit into the v4 migration model. Since the existing discussion does not specifically call out the case in which the absence of a filesystem is noted while attempting to cross into the absent file system, it makes it hard to understand how referrals would work within the existing protocol. This needs to be corrected to allow better understanding of the capabilities of NFSv4.0 which will be retained in future minor versions of NFSv4. It makes sense to present a description of referrals in a new sub-section following the "Migration" section, and would be section 6.2.1, given the current numbering scheme of RFC 3530. The material in the previous sub-sections of this document should be helpful in explaining the details of referral handling.

There are also a number of cases in which the existing wording of RFC 3530 seems to ignore the referral case of the migration feature. In the following specific cases, some suggestions are made for edits to tidy this up.

- In section 1.4.3.3, in the third sentence of the first paragraph, the phrase "In the event of a migration of a filesystem" is unnecessarily restrictive and having the sentence read "In the event of the absence of a filesystem, the client will receive an error when operating on the filesystem and it can then query the server as to the current location of the file system" would be better.

- In section 6.2, the following should be added as a new second paragraph: "Migration may be signaled when a file system is absent on a given server, when the file system in question has never actually been located on the server in question. In such a case, the server acts to refer the client to the proper fs location, using fs_locations to indicate the server location, with the existence of the server as a migration source being purely conventional."

- In the existing second paragraph of section 6.2, the first sentence should be modified to read as follows: "Once a filesystem has been successfully established at a new server location, the error NFS4ERR_MOVED will be returned for subsequent requests received by the server whose role is as

the source of the filesystem, whether the filesystem actually resided on that server, or whether its original location was purely nominal (i.e. the pure referral case)."

- The following should be added as an additional paragraph to the end of section 6.4, the: "Note that in the case of a referral, there is no issue of filehandle recovery since no filehandles for the absent filesystem are communicated to the client (and neither is the fh_expire_type)".

- The following should be added as an additional paragraph to the end of section 8.14.1: "Note that in the case of referral, there is no issue of state recovery since no state can have been generated for the absent filesystem."

- In section 12, in the description of NFS4ERR_MOVED, the first sentence should read, "The filesystem which contains the current filehandle object is now located on another server."

## 6. Acknowledgements

The author wishes to thank Neil Brown for his helpful comments.

## 7. Normative References

[RFC3530]
S. Shepler, et. al., "NFS Version 4 Protocol", Standards Track RFC

## Author's Address

David Noveck
Network Appliance, Inc.
375 Totten Pond Road
Waltham, MA 02451 USA

Phone: +1 781 768 5347
EMail: dnoveck@netapp.com

## Full Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78 and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on

an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.