

# NFSv3 to NFSv4 Migration Guide

Author: Vincent Roqueta

Contributors: Tony Reix

Draft – version 0.3 – 2005/12/06 – Bull

## I Introduction

NFSv4 is a network file system. NFSv4 is the successor of NFSv3. It has been designed to work on a LAN or over the Internet. As the Internet is a world-wide, unsecure, slow and heterogeneous network, NFSv4 comes with several new features:

- Advanced security management
  - Kerberos
  - SPKM
  - LIPKEY
- Firewall friendly
- Advanced and aggressive cache management
- Non Unix compatibility (Windows)
- Easy to administer (replication, migration)
- Crash recovery (Client and server sides)

## II External differences between NFSv3 and NFSv4

NFSv4 is not just a simple protocol evolution of NFSv3. It has been fully re-designed to overcome the limitations of its predecessors. As a consequence it works in a different way.

### ***1 Use of one network port***

To help NFS setup for internet use, one unique network port is used on NFSv4. This predetermined port is fixed. Default is port 2049.

#### **a) Consequences:**

- Using NFS through firewalls is easier.
- `portmapper` is no longer required.

#### **b) Restrictions :**

- Some security protocols may use other ports.

### ***2 NFSv4 virtual root directory***

NFSv4 uses the “virtual root directory” concept. In a similar way to other commonly-used Internet protocols (ftp, http, etc.), the goal is to create a “virtual root directory” that provides a unique access

from the network to the exported data, i.e., exported files and directories. This technique provides a better data control and thus enhances security.

This directory is exported by the server with the identifier `fsid=0`. The virtual root directory can contain symbolic links to other system directories.

On the client side, all directories which have to be mounted are seen from the virtual root directory, i.e., with the assumption that `/nfs/root/directory/` is exported from the server as the NFSv4 virtual root, mounting `server:/` on the client side means we try to access data stored in the server's `/nfs/root/directory/` directory.

#### **a) Consequences:**

- One unique virtual root directory is exported from the server with the identifier `fsid=0`.
- Data to be exported must belong to the virtual root sub path, i.e., be a subdirectory of the virtual root. It could be a symbolic link to system directory.
- Exporting two or more NFSv4 virtual roots (i.e., exporting several directories with the option `fsid=0`) will cause unexpected behaviour. Only one of these directories will be accessible through NFSv4 : the first export directory.
- Clients must ask to mount directories from the virtual root.

### **3 TCP / UDP**

In order to ensure a better reliability over the Internet, NFSv4 only uses TCP.

#### **a) Consequence**

UDP Protocol is not available.

#### **b) Restrictions**

RDMA (Remote Direct Memory Access) support for NFSv4 has been implemented. Support of other protocols may be added later.

### **4 Ipv6 Support**

IPv6 protocol is supported on the client side.

#### **a) Restrictions**

Using IPv6 requires a modified kernel. IPv6 is not yet supported on the NFSv4 server side.

### **5 32 KB blocs**

NFSv4 uses 32 KBytes pages.

#### **a) Consequence**

Better performances will be delivered when specifying a “`rsize`” and a “`wsize`” of 32 KBytes.

Default value of mount read and write is 1024 Bytes. See the section “Mounting”.

## **6 Integrated protocol**

NFSv4, unlike its predecessors, brings together in the same protocol most file system management tools. NFSv4 only requires the servers below to be launched :

- `rpc.nfsd` (nfs server)
- `rpc.idmapd` (identifier server)

### **a) Consequence**

Tools such as `rpc.lockd` or `rpc.quotad` are no longer required.

### **b) Restriction**

`rpc.mountd` is still needed. This will be changed later.

## **7 IDMapper**

In order to enhance, extend and connect the management of users to various tools (LDAP, security protocols, etc.), NFSv4 uses an independent identifier mapper : `rpc.idmapd`.

### **a) Consequences**

- To match user X on the client to the user Y on the server, use `/etc/idmap.conf`. It is advised to configure identifiers `nobody`, `nfsnobody` and `root`. A default configuration is provided.
- The idmapper can be linked to the directory LDAP and the identification server Kerberos.
- To use NFSv4, the `rpc.idmapd` server must be configured and must run on both client and server sides.

### **b) Restrictions**

Idmap and LDAP matching has not been tested yet.

## **8 No backward compatibility**

The NFSv3 and NFSv4 protocols are not compatible. A NFSv4 client cannot access a NFSv3 server, and vice versa. However, in order to simplify migrations from NFSv3 to NFSv4, both NFSv3 and NFSv4 services are launched by the command: `rpc.nfsd`.

In the case of NFSv3 and NFSv4 clients simultaneously accessing the same server, one must be aware that two different file systems are used: there is no backward support to NFSv3 by the NFSv4 server.

## III Migration

Migrating a system is a 5 step process:

### 1 *Listing data to export*

The first step is to list all data to be exported by NFSv3. Usually, there are several directories to be exported. In the example that follows, the following directories are used :

```
/home/user1/  
/home/user2/  
/var/data/data1/  
/var/data/data2/
```

### 2 *Choosing the NFSv4 virtual root*

To define the NFSv4 virtual root, you can either :

1. Create a NFSv4 root directory. Example: `/exports/` .
2. Choose an existing directory which will be used as virtual root for. Example: `/home/` .

In the following examples, `/exports` is used as the virtual root.

### 3 *Data migration to the virtual root*

Once we have a virtual root, we need to make data exported from NFSv3 appear in the NFSv4 virtual root, as defined previously.

There are three main possibilities:

- Copy or move data from its path to the virtual root (recommended):

```
mv /var/data/data1/ /exports/
```

- Create a symbolic link from the virtual root to the data path:

```
ln -s /var/data/data2/ /exports/data2
```

- Move the data path sub tree to the virtual root (recommended):

```
mkdir /exports/user1/  
mount -bind /home/user1/ /exports/user1/  
mkdir /exports/user2/  
mount -bind /home/user2/ /exports/user2/
```

At this point, all NFSv3 exported data appears in the `/exports` directory.

### 4 *Modifying export options*

The goal is to convert NFSv3 export options into NFSv4 ones. Most of the NFSv3 and NFSv4

options are the same. (NFSv4 requires an additional option to `exportfs` to provide the virtual root).

### a) Virtual root export

The first step is to export the virtual root. Generally NFSv3 options can be reused within NFSv4.

Be sure to add the option `fsid=0` when exporting the virtual root. This option indicates that the exported directory is the NFSv4 virtual root.

A typical NFSv4 export is :

```
exportfs -ofsid=0,insecure,no_subtree_check */exports
```

### b) Exporting sub-directories of the virtual root

The parameters given to `exportfs` for exporting sub-directories are identical for NFSv3 and NFSv4:

```
exportfs -orw,nohide,insecure,no_subtree_check */exports/user1
```

```
exportfs -orw,nohide,insecure,no_subtree_check */exports/user2
```

## 5 Mounting

Mounting remote share is done by using the following command :

```
mount -t nfs4 server:/user1 /my/mount/point
```

Note that the paths provided within a `mount` command are related to the virtual root, and not to the real server root.

So, using the previous configuration and examples, the following commands:

```
mount -t nfs4 server:/ /mnt/nfs_server
```

```
ls /mnt/nfs
```

will list the data in the server virtual root, i.e. :

```
user1, user2, data1, data2.
```

To enhance performance, specify 32 Kbytes `rsize` and `wsize` :

```
mount -t nfs4 server:/ /mnt/nfs_server -orsize=32768,wsize=32768
```

## **IV Known bugs and limitations:**

Known bugs and limitation, November 20, 2005.

### **1 *Functional limitations***

#### **a) All platforms**

- SPKM, LIPKEY are not supported.
- Migration and replication are not supported.
- Named attributes are not supported.
- “Mandatory” locks do not conform with the expected behaviour (but POSIX advisory locks do work).
- Number of ACL rules is limited to 35 entries (Oops after 35).
- `mountd` is mandatory and a `portmapper` must be running (`rpcbind` or `portmapd`, for example).

#### **b) NFS – LUSTRE / GPFS interoperability**

- NFS is not able to export a LUSTRE file-system.
- Locks propagation over GPFS does not work.

### **2 *Performance limitations***

- The delegation mechanism does not use statistics to enhance performance. Delegation is given to all client supporting this option.
- NFSv4 is at least as fast as NFSv3, but it is not yet optimised.

Despite these limitations, NFSv4 is more robust than NFSv3 and performs better on LANs, either with or without Kerberos being activated.