NFS-GANESHA : Users Guide

18 March 2008

Started: 2008.03.18

Philippe Deniel: philippe.deniel@cea.fr
Thomas Leibovici : thomas.leibovici@cea.fr

1. Introduction

2. Getting started

The configuration file for GANESHA may look huge and complex at first glance. In fact, it has a very modular structure which reflects the overall layered architecture of the project itself. The configuration file is made of several blocks; each block is related to the configuration details about a given feature.

A block will look like this:

```
<BLOCK_TAG>
{
<Config_Item_Tag> = <Value> ;
# Comment
<Config_Item_Tag> = <Value>
    .
    .
<Config_Item_Tag> = <Value> ;
}
```

There are different blocks and blocks type available, most of them are not mandatory but using them will provide you with finer control to the NFS daemon's behavior.

Block are made of keys associated with values, each association representing a configuration parameter. In later section, we will tell you more about each block and each configuration item, but for now, let's focus on the minimal information to be set in the configuration file to get starting. For the moment, we will talk only about the mandatory blocks, for each use of GANESHA (POSIX/PROXY/SNMP/FUSELIKE).

2.1.First required block: the EXPORT block

This is an example of an export block in the GANESHA configuration file. One such block should exist per export entry made in the NFS daemon. An EXPORT block is equivalent to a line with /etc/exports for the nfsd within Linux

```
EXPORT
  {
    Export_Id = 1 ;

  Path = "/my/path/to_export" ;
    Root_Access = "client1,client2,localhost";
    Access = "myclients.net";

    Pseudo = "/nfsexport/ExportID1";

    Access_Type = RW ;
      Anonymous_root_uid = -2 ;

  NFS_Protocols = "2,3,4" ;
      Transport_Protocols = "udp,tcp" ;
      SecType = "sys" ;


    MaxRead = 8192 ;
      MaxWrite = 8192 ;
      PrefRead = 8192 ;
      PrefWrite = 8192 ;
```

```
        PrefReaddir = 8192 ;

     Filesystem_id = 100.1 ;

  Cache_Data = TRUE ;

     FS_Specific = "" ;

        Tag = "ganesha" ;
 }
```

As you can see there are several possible configuration items within this block. Let's see what they mean.

2.1.1.How to describe clients ?

Clients can be single machines, pack of machines, netgroups, and networks:
- Single machine is identified by its name or ip address: e.g. localost or 127.0.0.1
- Pack of machine: you may have "farms" of machines with similar name like "cluster0, cluster0, ..., cluster-gateway". With a cluster, you can have bunches of such names. Using "unix jokers" can help so far. GANESHA allows you to specify names like "cluster*" or "cluster1?", using the old style "shell joker syntax"
- Networks: this should be a network name (you may have a /etc/networks file describing them) or a reduced ip address describing addresses belonging to this networks. For example 12.13.0.0 will describe machines whose address is 12.13.x.y with netmask 0xffff0000
- Netgroups: clients can be identified by the fact they belong to a given netgroup. This is shown by adding an arobas at the beginning of the netgroup's name. To tell GANESHA to provide access to machine in netgroup "nfsclients" specify "@nfsclients"

Lists of clients is comma separated: "localhost,12.13.0.0,@nfsclient"

2.1.2.Other Configuration item in the EXPORT block

Only 'Path' , 'Export_Id' and 'Pseudo' are mandatory keys.

This are the available configuration items:
- Export_Id : This tag is used to set the the id for this export. This is mostly used internally, but this is a mandatory value. The value is to be a non-zero integer value. Each EXPORT block **must** have a different Export_Id.
  - *e.g: Export_Id = 1 ;*
- Access: The list if clients that can access the export entry.
  - *e.g: Access = "Machine3,Machine10*,NetworkB,@netgroupY";*
    - NB: Using Access = "*" ; will allow everybody to access the
    - mount point.

- Root_Access: The list of clients (see above) that have root access
    - rights (root remains root when crossing the mount point).
    - e.g: Root_Access = "localhost,12.13.0.0,@nfsclient" ;
- Access_Type: Describes the kind of access you can have of the mount point. Acceptable values are:
  - RO: Read-Only mount point
  - RW: Read-Write mount points ;
  - MDONLY: The mount point is RW for metadata, but data accesses are forbidden (both read and write). This means you can do everything on md, but nothing on data.
  - MDONLY_RO: Like RO, but reading data is forbidden. You can read only metadata.
    e.g: Access_Type = "RW" ;
- Anonymous_root_uid : The uid to be used for root when no root access was specified for this clients. This is the definition of the 'nobody' user. "Traditional" value is -2
  - *e.g: Anonymous_root_uid = -2 ;*
- NOSUID: a flag (with value TRUE or FALSE) showing if setuid bit is kept.
  - *e.g : NOSUID= TRUE;*
- NOSGID: a flag (with value TRUE or FALSE) showing if setgid bit is kept.
  - *e.g : NOSGID= TRUE;*
- NFS_Protocols: The version(s) of the NFS protocol you can use for mounting this

export entry

- o *e.g: NFS_Protocols = "2,3,4" ;*
- Transport_Protocols: The transport layer to use for mount this entry. This should be UDP or TCP or a list.
  - o *e.g : Transport_Protocols = "UDP,TCP" ;*

- Sec_type: List of supported RPC_SEC_GSS authentication flavors for this export. It can be a coma-separated list of the following values: sys, krb5i, krb5p.
  - o *e.g : SecType = "sys,krb5";*
- MaxOffsetRead: Maximum offset allowed for a read operation (no limit if not specified). This could be seful for preventing "evil" use of NFS read (like access a TB long file via NFS)
  - o *e.g : MaxRead = 409600;*
- MaxOffsetWrite: Like MaxOffsetRead, but for Write operation
  - o *e.g : MaxWrite = 409600;*
- MaxRead, MaxWrite, PrefRead, PrefWrite, PrefReaddir: The value to be eturned to client when NFS3_FSINFO is called. Better solution is not to use his keys and so keep the default values, optimized for NFSv3.
- Filesystem_id: The filesystem id to provide to the client for this xport entry. NFS Client will use this value to address their nternal metadata cache. In NFSv4, both major and minor values re used, in NFSv2 and NFSv3, only the major is used.
  - o *e.g. : Filesystem_id = 100.1 ;*
- PrivilegedPort: A flag (TRUE or FALSE) to specify is a client using an ephemeral port should be rejecting or not.
  - o *e.g. PrivilegedPort = FALSE ;*
- Cache_Data: A flag (TRUE or FALSE) To specify if files content should be cached by the GANESHA server or not
  - o *e.g : Cache_Data = TRUE ;*
- MaxCacheSize: if export entry is datacached, this value defines the maximum size of the files stored in this cache.
- FS_Specific: a comma separated list of information used by the FSAL module (see later) to perform initialization. See FSAL documentation for detail.
- Path: The path to export via NFS. Should have a leading slash.
  - o *e.g: Path = "/nfs/my_exported_directory" ;*
- Tag: A way of providing a shorter path for mounting an entry. For example, you could mount an entry like this:

mount -o vers=3 nfsserver:/nfs/my_exported_directory /mnt

But if you specified "Tag = ganesha;", you can simply do

mount -o vers=3 nfsserver:ganesha /mnt

- Pseudo: a NFSv4 specific key that shows the path, in NFSv4 pseudo file system were the 'actual' mount point resides.
    - *e.g : Pseudo = "/nfsv4/pseudofs/nfs_mount_entry_#1" ;*

2.2.Second required step: configuring the FSAL

GANESHA is capable of managing different kinds of namespaces. For each supported namespaces, a dedicated module, called FSAL (which stands for *File System Abstraction Layer*) is used. Each FSAL module has dedicated configuration block and items

2.2.1.Configuring the POSIX FSAL

GANESHA NFS server makes it possible to export any filesystem where entries can be accessed using handles.
In a POSIX filesystem, entries are accessed using their path, which does not meet the requirements for a handle (persistency, unicity). However GANE-SHA provides a FSAL that can assign a unic and persistent filelehandle to each filelesystem entry. To do this, it needs to keep some persistent information in a database.

GANESHA's FSAL POSIX supports both PostgreSQL versions 7 and 8.
This section will explain how to install/configure a PostgreSQL 8.1 database in order to use the POSIX FSAL. For 7.x version, configuration is very similar (differences will be noticed inline).
In the following description, replace *%DBNAME%* and *%USERNAME%* signs with the actual database name and user name you want to use.

- First, install the postgreSQL 8.1 package.
- Then, take the "postgres" identity (this user is created during package setup. It has all accesses granted on PostgreSQL engine)
- su - postgres
- Create a new user for using the database with GANESHA:

```
        createuser   --no-superuser   --no-createdb   --no-createrole
        --login --pwprompt %USERNAME%(you will be prompted for a password).
```
With postgreSQL 7, use the following command instead:
```
        createuser --no-adduser --no-createdb --pwprompt %USERNAME%
```
(answer 'no' to questions that will be prompted, and enter a password)

- Create a new database (owned by the user we have juste created):
  ```
          createdb -O %USERNAME% %DBNAME%
  ```
- In order to use PGSQL Procedural Language for improving frequent database queries, we have to activate plpgsql into our database:
  ```
          createlang plpgsql %DBNAME%
  ```

Make sure you have tcp connections enabled for your database. This is set in file *postgresql.conf* (it should be located in `/var/lib/pgsql/data'). Make sure *tcpip_socket* parameter is true and the line is not commented for PGSQL v7:
```
        tcpip_socket = true
```

In order to enable server's authentication, you have to modify *pg_hba.conf* (by default, this is located in the /var/lib/pgsql/data directory).

At the end of the file you should have something like this:

```
        local all all trust
        # IPv4 local connections:
        host all all 127.0.0.1/32 md5
        host all all %GANESHA_NFSD_IP%/32 md5
        # IPv6 local connections:
        host all all ::1/128 trust
```

After this step, you have to restart the postgresql service:
```
        service postgresql restart
```

We can now create the tables in the database. To do this, retrieve the appropriate SQL script from directory *src/FSAL/FSAL_POSIX/DBExt/PGSQL* in GANESHA source tree: use *posixdb_v7.sql* if you have a postgreSQL v7.x database, *posixdb_v8.sl'* if you are using postgreSQL v8 database or higher version (with stored procedures support).

Then execute it like this:
```
        cat posixdb_v8.sql | psql -h localhost -U %USERNAME% %DBNAME%
```

Create a keytab file in order for GANESHA to access the database. The content of this file must have the folowing syntax:

```
hostname:port:database:username:password
```

Example of DB keytab's content:
```
localhost:5432:GANESHA_FS_DB:GANESHA_DB:ganesha
```

Take care of setting exactly the same values in the GANESHA's configuration file for DB_host, DB_port, DB_name and DB_login. This file's permissions MUST be 600 (rw   -).

2.2.2.The POSIX FSAL in GANESHA's configuration file

With POSIX FSAL, you will need to define a block with the tag "POSIX". If you use POSIX FSAL, then you have already setup the prerequisite PostgreSQL database (See the document "NFS_GANESHA: Configuration of the FSAL POSIX"). This means that you have configure a machine to run the DB, listening on a well known port. You have already setup a DB table to be used by a DB user and wrote a DB keytab for him.

The item to be used are the following ones:

- DB_Host : The hostname for the machine running the DB engine
  e.g. : DB_Host = "dbserver.localdomain" ;
- DB_Port : The port to use for contacting the DB engine
  e.g. : DB_Port = 5432;
- DB_Name : The name for the database to use for the POSIX/FSAL
  e.g. : DB_Name = FSAL_POSIX_DB ;
- DB_Login : The username to use to connect to the database
  e.g.: DB_Login = DB_USER ;
- DB_keytab : The keytab to use to acquire the 'DB_Login' identity.
  e.g : DB_keytab = "/var/etc/posix.db.keytab" ;

Example of POSIX block :
```
POSIX
{
    DB_Host = "dbserver";
    DB_Port = 5432;
    DB_Name = DEMO_DB ;
    DB_Login = DB_USER ;
    DB_keytab = /tmp/posixdb.keytab ;
```

```
}
```

2.2.3.The SNMP FSAL in GANESHA's configuration file

In SNMP (Simple Network Management Protocol), data are organized as a tree
where all objects are addressed by their OID, which is the object's path in this tree. For
example, ``.iso'' can be considered has a directory
identified by the OID ``.1'', and the leaf ``.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0''
(which corresponds to system description) can be considered as a file
whose data is something like ``Linux node_name 2.6.9-22.ELsmp #1 SMP Mon Sep 19
18:32:14 EDT 2005 i686''.

As a result, exporting such a tree through NFS makes it possible to have a namespace
similar to ``/proc'', where administrators can read statistics about a system or an
equipment (switch, router, ...) simply using cat command, and modify them easily, using
vi or echo "xxx" > file.

For configurating GANESHA's SNMP access, you have to set some options in the
configuration file: this in done in a ``SNMP *configuration block*.

In such a block, you can set the following values:

- snmp_version: this indicates the SNMP protocol version that GANESHA will use
  for communicating with SNMP agent. Expected values are **1**, **2c** or **3**. Default is
  2c.
- snmp_server: this is the address of the SNMP master agent. A port number can
  also be specified by adding **:<port>** after the address. Default is **localhost:161**.
- nb_retries: number of retries for SNMP requests. Default value is
  SNMP_DEFAULT_RETRIES, defined in net-snmp library.
- microse_timeout: number of microseconds until first timeout, then an exponential
  backoff algorithm is used for next timeouts. Default value is
  SNMP_DEFAULT_TIMEOUT, defined in net-snmp library.
- client_name: this is the client name that could be used internally by net-snmp for
  its traces. Default value is **GANESHA**.
- snmp_getbulk_count: this indicates the number of responses wanted for each

SNMP GETBULK request. Default value is **64**.

SNMP v1 and v2 specific parameters:

- community: this is the SNMP community used for authentication. Default is **public**.

SNMP v3 specific parameters:

- auth_proto: the authentication protocol (MD5 or SHA). Default is **MD5**.
- enc_proto: the privacy protocol (DES or AES). Default is **DES**.
- username: the security name (or user name). This a private information: check rights on this config file!
- auth_phrase: authentifaction passphrase (>=8 char). This a private information: check rights on this config file!
- enc_phrase: authentifaction passphrase (>=8 char). This a private information: check rights on this config file!

A simple SNMP v2c example:

```
SNMP
{
    snmp_version = 2c;
    snmp_server = "snmp_master.my_net";
    community = "public";
}
```

A simple SNMP v3 example:

```
SNMP
{
    snmp_version = 3;
    snmp_server  = "snmp_master.my_net";
    username     = "snmpadm";
    auth_phrase  = "p4ssw0rd!";
    enc_phrase   = "p455w0rd?";
}
```

Export entries

For defining the path of an export entry, you must replace traditional SNMP dot separators '.' by slashes '/'. For example, you shouldset export path to '/iso/org' instead of '.iso.org'.

Note that you can give slash separated numerical OIDs for exports. Thus, exporting

'/1/3/6/1/2/1' is equivalent to '/iso/org/dod/internet/mgmt/mib-2'.

Example of use:

```
$ cd /mnt/iso/org/dod/internet/mgmt/mib-2
$ cat host/hrSystem/hrSystemUptime/0
528224338 (61 days, 03:17:23.38)
$ cat udp/udpInDatagrams/0
820798
$ cat udp/udpTable/udpEntry/udpLocalAddress/1/0/0/127/123
127.0.0.1
```

2.2.4.Using PROXY FSAL

Things are relatively easy to configure with the configuration of the PROXY FSAL. Roughly, this FSAL operates as a NFSv4 client whose API is the one of the FSAL. In this paragraph, the terminology *server* will be used to identify the NFSv4 server accessed by GANESHA as a client via PROXY FSAL. NB: If you use PROXY FSAL, then only NFSv4 EXPORT will be supported, the daemon will refuse to export stuffs in NFSv2 or NFSv3 (GANESHA is basically a NFSv4 PROXY in this case).

For the setup of the PROXY FSAL, you will use a block with tag *NFSv4_Proxy*. Its configuration items are:
- Srv_Addr: the hostname or IP address of the NFSv4 server to be accessed (mandatory)
- Retry_SleepTime: number to seconds to wait for reconnection to the server in case of server lost connection. The default value is 60 seconds
- NFS_Proto: the transport protocol to be used. This parameter should have value "tcp" or "udp.

The following items are optional. If they are not defined in the configuration file, their default values will be used.
- NFS_Port: the port used for NFS export. You can omit this parameter and then use the default value of 2049.
- NFS_Service: the rpc service number to be used for NFS. . You can omit this parameter and then use the default value of 100003.

- NFS_SendSize, NFS_RecvSize: the size for the RPC buffer used. I strongly recommend not to use this parameter and then keep the default value of 32kB. Anyway, for performance tuning, it may be interesting to use other values.

Example:

```
NFSv4_Proxy
{
        Srv_Addr = mynfsv4server ;
        NFS_Port =    2049 ;
        NFS_Service = 100003 ;
        #WARNING /!\  Small NFS_SendSize and NFS_RecvSize may
        lead # to problems
        NFS_SendSize = 32768 ;
        NFS_RecvSize = 32768 ;
        Retry_SleepTime = 60 ;
         NFS_Proto = "udp" ;
         #NFS_Proto = "tcp" ;
}
```

2.2.5. Exporting FUSE-based file systems

2.2.5.1. Quick Start

Compiling and installing

Go to the "src" directory of NFS-GANESHA distribution. Then run:

```
./configure --with-fsal=FUSE
make
make install
```

Exporting your FUSE-based file system

● In your source code, replace **#include <fuse.h>** with **#include <ganesha_fuse_wrap.h>**

● For linking your program, replace **-lfuse** with **-lganeshaNFS**

That's done! You can now start your daemon and mount your filesystem

(here is an example in NFSv3):

```
./my_daemon
mount -o vers=3,udp localhost:/ /mnt
```

2.2.5.2. Details about GANESHA FUSE-like binding

Interface

Ganesha FUSE-like interface provides most FUSE's "high-level" structures and calls (struct fuse_file_info, struct fuse_operations, fuse_main(), fuse_get_context(), ...)

Basically, it supports FUSE-based filesystems using <fuse.h> (at least version 2.6).

Filesystem mandatory features

For being able to export your filesystem with NFS-GANESHA, the following features are mandatory:

- **getattr**must be implemented
- Each entry in your filesystem must have a unic <st_ino, st_dev> peer
- You must set a correct value to the "st_mode" field of "struct stat" (type and access mode)
- The "st_nlink" field of "struct stat" must not be null
- Deprecated call "getdir" is not supported, replace it with "readdir"

Command line arguments

fuse_main() parameters slightly differ from FUSE implementation.

The expected command line parameters are:

```
Usage: fusexmp [-hds][-L <logfile>][-N <dbg_lvl>][-f
<config_file>]
       [-h] display this help
       [-s] single-threaded (for MT-unsafe filesystems)
       [-L <logfile>]      set the logfile for the daemon
       [-N <dbg_lvl>]      set the verbosity level
       [-f <config_file>]  set the config file to be used
       [-d]                the daemon starts in background, in a
new process group
```

```
        [-R]                daemon will manage RPCSEC_GSS (default
is no RPCSEC_GSS)
        [-F] <nb_flushers>  flushes the data cache with purge, but
do not answer to requests
        [-S] <nb_flushers>  flushes the data cache without purge,
but do not answer to requests
```

Example

FUSE-binding examples are provided in the GANESHA repository (directory src/example-fuse).

These are the same as provided with FUSE distributions, except that "#include <fuse.h>" have been changed to "#include <ganesha_fuse_wrap.h>".

After compiling GANESHA with FUSE FSAL, you can simply run it the following way:

```
./fusexmp
```

3. Using the Data Cache

If you want to use the data cache (this can be very interesting in some situation like running a NFS PROXY), you'll have to configure a few things in the configuration file.

Cache Data is enabled per export entry, via the *Cache_Data* item in the Export block. If value is TRUE, then the data cache will be used.

Two blocks are required to configure the data cache: the *FileContent_Client* block and the *FileContent_GC_Policy* block. Both of them have various item no most of them are optionnal and their default values will be enough for basic use. Anyway, some items needs to be set.

In FileContent_Client, use the *Cache_Directory* item to set the physical location of the cache. This have to be a regular filesystem. Most of the situation will be deal with a simple block like this one:

```
FileContent_Client
{
        Cache_Directory = /local/ganesha.datacache ;
```

```
    }
```

In FileContent_GC_Policy, you will give argument to drive the behavior of the cache's garbage collection and file's retentions within the cache.

The items to be used are these ones:

- Emergency_Grace_Delay: When doing a global datacache flush (option -F of the ganesha), files who are younger than this delay will neither be flushed nor removed from the cache.

- Lifetime: the minimum delay (in seconds) a file has not been accessed, for beeing a candidate to flush and removal from the datacache. A value of "-1" disables data flushing.

- Df_HighWater: When the local datacache filesystem usage is over this value (in percent), a garbage collection of the datacache is launched.

- Df_LowWater: A datacache garbage collection stops when the filesystem usage falls to this value (in percent).

- Runtime_Interval: The interval between checking datacache filesystem usage.

Example:

```
    FileContent_GC_Policy

    {

# Lifetime for a file before being a candidate to GC

# A value of -1 will disable file GC

Lifetime = -1 ;


# The duration of inactivity needed for a file to become

# eligible for being fushed to the FSAL

# A value of -1 will disable file flush to FSAL

Inactivity_Before_Flush = 12000 ;
```

```
    # GC High Water Mark (in percent)

    Df_HighWater = 90 ;


    # GC Low Water Mark (in percent)

    Df_LowWater = 80 ;


    # Runtime interval (in seconds)

    Runtime_Interval = 12000 ;


    # Grace delay before flushing files

    Emergency_Grace_Delay = 600 ;
}
```

Manual flush: add "-F <nb_flusher>" to the ganesha command line to start a separated GANESHA process that will flush items from the data cache (without interfering with the eventually running NFS daemon).