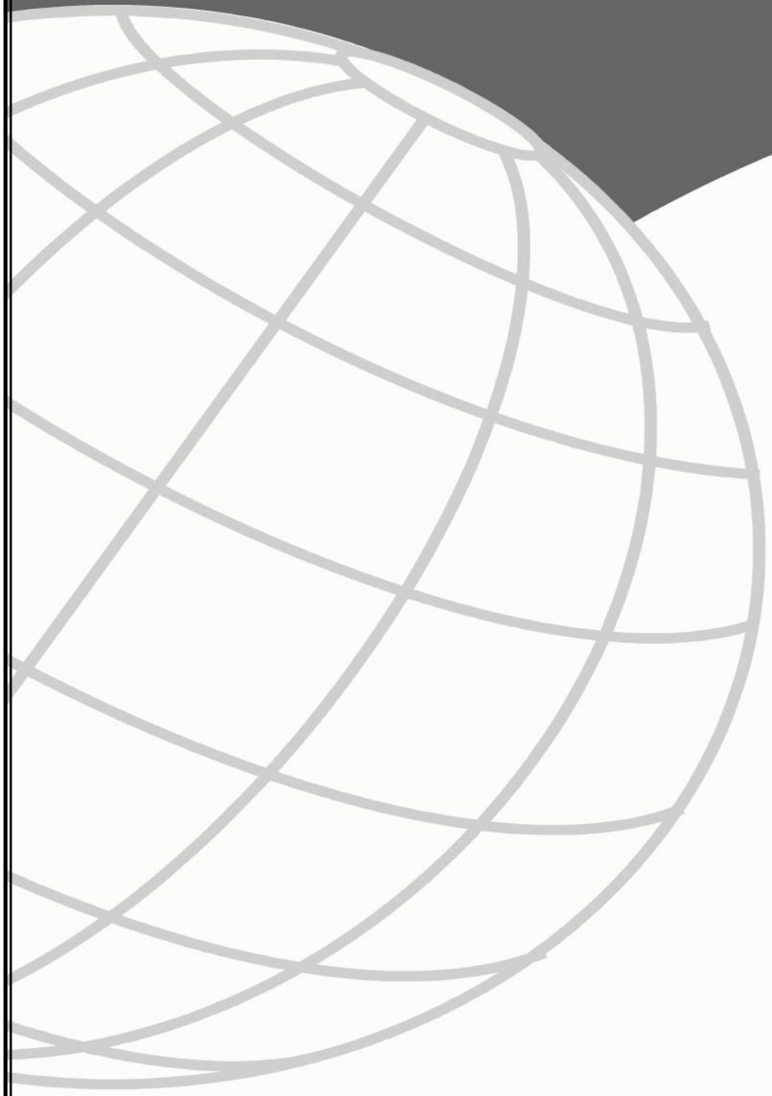


Enabling Service Excellence

Monitoring Network File Systems

eG Enterprise v6

Restricted Rights Legend

The information contained in this document is confidential and subject to change without notice. No part of this document may be reproduced or disclosed to others without the prior permission of eG Innovations Inc. eG Innovations Inc. makes no warranty of any kind with regard to the software and documentation, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Trademarks

Microsoft Windows, Windows NT, Windows 2000, Windows 2003 and Windows 2008 are either registered trademarks or trademarks of Microsoft Corporation in United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Copyright

Table of Contents

Table of Figures

Chapter

1

Introduction

NFS, or the Network File System (NFS), provides remote access to shared file systems across networks. Designed to be machine, operating system, network architecture, and transport protocol independent, NFS enables the export or mounting of directories to other machines, either on or off a local network. These directories can then be accessed as though they were local.

NFS uses a client/server architecture and consists of a client program, a server program, and a protocol used to communicate between the two. The server program makes filesystems available for access by other machines via a process called *exporting*. File systems that are available for access across the network are often referred to as *shared* file systems.

In environments spanning multiple private networks, the NFS plays a significant role in making critical file systems accessible to users across networks. Since users expect to access these *shared* file systems just as swiftly and effortlessly as they would the local ones, even the slightest of access delays can put him/her off. To ensure that the user experience with NFS remains pleasant, the client-server interaction of NFS should be continuously monitored.

eG Enterprise provides distinct monitoring models for monitoring the NFS server and client on Solaris and Linux, which measure the effectiveness of the server program as well as the experience of the client. This document provides the details of all these models.

Chapter

2

Monitoring NFS on Solaris Servers

To monitor NFS on a Solaris server, eG Enterprise provides the *NFS Solaris server* monitoring model (see Figure 2.1). This model monitors the Connection and Connectionless RPC calls that were received by the NFS server from the NFS clients, reveals call failures, and provides accurate pointers to the root-cause of the failure, so that the administrators can initiate the required remedial action.

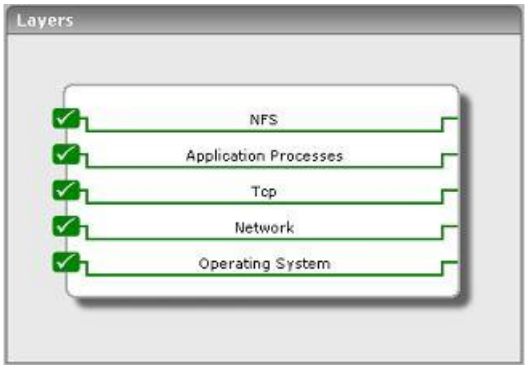


Figure 2.1: The layer model of an NFS Solaris server

Since the 4 layers at the bottom of Figure 2.1 have been dealt with extensively in the *Monitoring Generic Servers* document, let us proceed to look at the **NFS** layer alone.

2.1 The NFS Layer

This layer monitors the RPC calls received by the NFS server, and reports bad calls (if any) (see Figure 2.2).

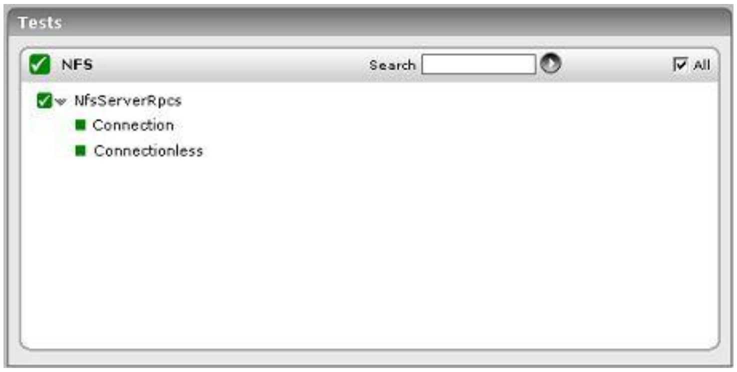


Figure 2.2: The test associated with the NFS layer

2.1.1 NFS Server RPCs Test

The NfsServerRpcs test reports the statistical information about the Connection and Connectionless RPC calls received by an NFS server. This test is applicable to Solaris OS only.

Purpose	Reports the statistical information about the Connection and Connectionless RPC calls received by an NFS server		
Target of the test	NFS on Solaris server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured.		
Outputs of the test	One set of results each for connection and connectionless RPC calls		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total number of calls: The total number of RPC calls received by the server during the last measurement period.	Number	This metric is a measure of the server workload.
	Number of badcalls: The total number of calls rejected by the RPC layer (the sum of badlen and xdrcall as defined below) during the last measurement period.	Number	Ideally, there should be very few bad calls. If there are any bad calls, possible reasons could be authentication problems caused by having a user in too many groups, attempts to access exported file systems as the root user, or an improper secure RPC configuration.

	Number of nullrecv: The number of times an RPC call was not available when it was thought to be received.	Number	Too many null receipts can indicate that NFS requests are not arriving fast enough to keep all nfsd daemons busy. Consider reducing the number of NFS server daemons until null receipts are reported.
	Number of badlen: The number of RPC calls in the last measurement period with a length shorter than a minimum-sized RPC request (i.e. corrupt RPC requests).	Number	This metric indicates malformed NFS requests that can be caused by bugs in the client or server software of by physical network problems.
	Number of xdrcall: The number of RPC calls in the last measurement period whose header could not be XDR decoded.	Number	This metric indicates malformed NFS requests that can be caused by bugs in the client or server software or by physical network problems.
	Number of dupchecks: The number of RPC calls in the last measurement period that looked up in the duplicate request cache.	Number	The duplicate request cache keeps a record of previously executed NFS requests. The dupchecks value reports the number of times this cache was consulted or checked.

	Number of dupreqs: The number of RPC calls in the last measurement period that were found to be duplicates.	Number	The dupreqs count indicates the number of times a check of the duplicate request cache had a "hit" – i.e. the number of times the NFS server received a previously executed request. For connection-oriented requests, a high dupreqs to dupchecks ratio is 0.01%. For connectionless requests, a high ratio of dupreqs to dupchecks is 1%. High ratios indicate one of three problems: ▪ The timeout set on one or more clients' NFS mounts is too low: Adjust the *timeo* option in the automounter map or the NFS mount command upward. ▪ The server is not responding quickly enough: There could be lots of reasons for this having to do with physical capabilities of the server, such as, processor speed, numbers of processors (if it is a multiprocessor), not enough primary memory (check if the percentage of reads is high, say over 5%; this would indicate lots of reads that would be best served from cache if there was enough memory), numbers of disk drives on the system (spreading more data accesses across more spindles reduces response time; if you've eliminated primary memory as a cause, check if the percentage of writes is high, say over 5%), etc. Other possibilities extend to artificial limits, such as the number of server threads set via nfsd.

			- There is a routing problem impeding replies from the server to one or more clients.

Chapter

3

Monitoring NFS on Solaris Clients

To monitor NFS on a Solaris client, eG Enterprise provides the *NFS Solaris client* monitoring model (see Figure 3.1). Besides monitoring the connection and connectionless RPC calls that were initiated by the client and accurately isolating the reason for call failures, this model also reveals how quickly the clients were able to access the *shared* directories.

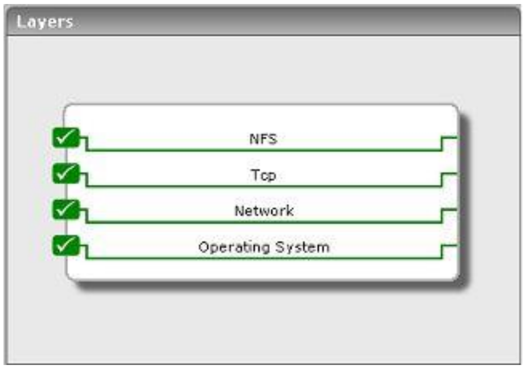
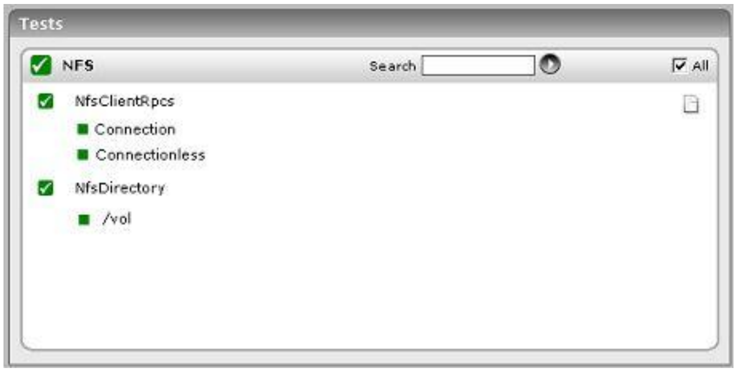


Figure 3.1: Layer model of an NFS Solaris client

Since the 3 layers at the bottom of Figure 3.1 have been dealt with extensively in the *Monitoring Generic Servers* document, let us proceed to look at the **NFS** layer alone.

3.1 The NFS Layer

The tests associated with this layer monitors the connection and connectionless RPC calls that were initiated by the client and accurately indicates the reason for call failures. In addition, the layer also reveals how quickly the clients were able to access the *shared* directories (see Figure 3.2).



Tests

☑ NFS Search All

☑ NfsClientRpcs
- 🟩 Connection
- 🟩 Connectionless

☑ NfsDirectory
- 🟩 /vol

Figure 3.2: The tests associated with the NFS layer

3.1.1 NFS Client RPCs Test

This test reports the statistical information about the Connection and Connectionless RPC calls made by the NFS client. The test is applicable to Solaris OS only.

Purpose	Reports the statistical information about the Connection and Connectionless RPC calls made by the NFS client		
Target of the test	NFS on Solaris client		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured.		
Outputs of the test	One set of results each for connection and connectionless RPC calls		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total number of calls: The total number of RPC calls made by the client during the last measurement period	Number	

	Number of badcalls: The number of times that an RPC call failed due to an error such as a timeout or an interrupted connection during the last measurement period.	Number	A non-zero value indicates timeouts or retransmissions. If a server has crashed, bad calls can be expected to happen. But, if bad calls happen during normal operation, then soft-mounted file systems use larger *timeo* value or a larger *retrans* value to avoid RPC failures. Note that on soft-mounted file systems, a request is retransmitted a limited number of times before it is reported as a failed RPC call. The value of badcalls is only incremented for the final failed attempt; previous failures increase the value of retrans. All requests that fail due to a timeout are recorded in timeouts.
	Number of badxids: The number of responses from servers for which the client has already received a response.	Number	If a client does not receive a response to a request within a time period, it retransmits the request. It is possible that the server may service the original request. In such a case, the client receives more than one response to a request. The value of badxid is incremented for every unexpected response. If the value of badxid is approximately equal to retrans, one or more servers probably cannot service client requests fast enough. Increase the *timeo* parameter for the NFS mount to alleviate request retransmission or tune the server to reduce the average request service time. With a large timeout count, if badxids are reported, it indicates that the network is dropping parts of NFS requests or replies. Reduce the NFS buffer size using the *rsize* and *wsize* mount parameters to reduce the probability of NFS buffer corruption during transmission.
	Number of timeouts: The number of calls that timed out waiting for response from a server during the last measurement period.	Number	If greater than 5% of all calls timeout, either the requests are not reaching the server or the *timeo* setting is too low. Check the badxids value to find the reason for timeouts.

	Number of newcreds: The number of times authentication information had to be refreshed during the last measurement period.	Number	
	Number of badverfs: The number of times the call failed due to a bad verifier in the response. This is a maintenance command.	Number	
	Number of timers: The number of times the calculated time-out value was greater than or equal to the minimum specified timeout value for a call. This is a maintenance command.	Number	
	Number of cantconn: The number of requests made by the client that could not connect to the server during the last measurement period. This is specific to connection based RPC calls.	Number	If greater than 1% of the total calls cannot connect, there is usually an NFS problem. Often, this is because the NFS server is down. It can also indicate that the connection queue length in the NFS server is too small, or that an attacker is attempting a denial of service attack on the server by clogging the connection queue. If the queue length is too small, use the –l parameter to nfsd to increase the queue length.
	Number of nomem: The number of times the call failed due to a failure to allocate memory. This is a maintenance command.	Number	
	Number of interrupts: The number of interrupted requests to a server by a client. This is specific to connection based RPC calls.	Number	

	Number of retrans: The number of repeated requests by the client to the server. This is specific to connectionless RPC calls.	Number	
	Number of cantsend: The number of requests that could not be sent by client to the server. This is specific to connectionless RPC calls.	Number	

3.1.2 NFS Directory Test

This test reports statistics relating to NFS file systems remotely mounted by a client. This test auto discovers the remote file systems on a client and periodically accesses the file systems to check their availability and access times.

Purpose	Reports the availability and access time for NFS file systems remotely mounted by a client		
Target of the test	NFS on Solaris client		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured.		
Outputs of the test	One set of results for every remotely mounted NFS		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Availability of the NFS files systems	Number	If the value of this measure is 0, it indicates that the file system is unavailable. The value 100 indicates the availability of the file system.

	Access time: Access time for the remotely mounted NFS file systems	Secs	By monitoring this value over time, an administrator can determine periods when NFS access is slow.

Chapter

4

Monitoring NFS on Linux Servers

To monitor Network File Systems on Linux servers, the eG Enterprise system offers a *NFS Linux server* monitoring model.

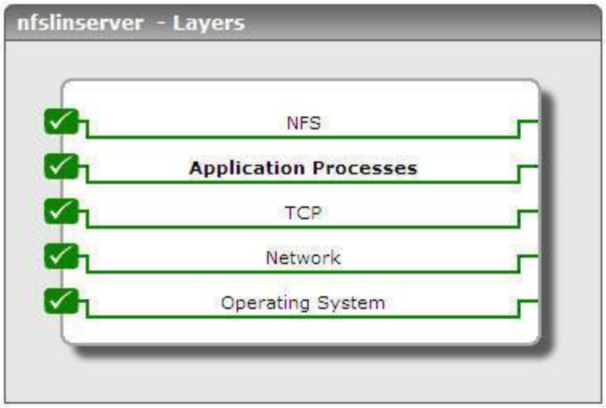


Figure 4.1: The NFS Linux server monitoring model

This model monitors the RPC calls received by the NFS, captures call failures, and also hints at what could have caused such failures, thereby enabling administrators to quickly find answers to the following questions:

- Is the NFS server available over the network?
- Has the server been sized to adequate CPU, memory, and disk space?
- Is TCP connectivity to the server good?
- Are all critical server processes up and running?
- Are clients able to communicate with the server, or is any critical TCP/UDP port of communication unavailable on the server?
- Is the RPC request load on the server very high?
- Have any RPC requests to the server failed?

- Were any corrupted RPC requests noticed?

The sections that will follow discuss the first layer of Figure 4.1 as other layers have already been discussed in the *Monitoring Unix and Windows servers* document.

4.1 The NFS Layer

The tests mapped to this layer monitor the RPC requests to the server and points to the error-prone requests. In addition, layer also runs availability checks on one/more configured TCP/UDP ports on the server.

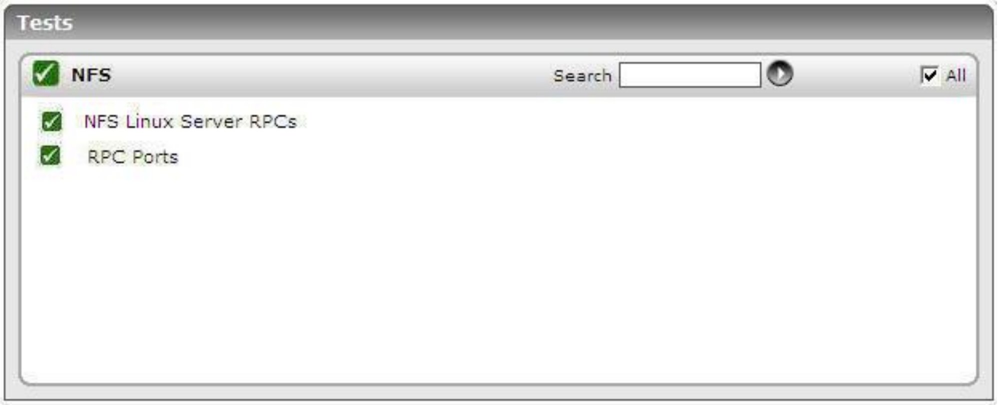


Figure 4.2: The tests mapped to the NFS layer

4.1.1 NFS Linux Server RPCs Test

NFS relies on *Remote Procedure Calls* (*RPC*) between clients and servers. Bad RPC or failure/corruption of RPC calls may result in clients being unable to access the shared file systems. Using this test, administrators can closely monitor the RPC calls and promptly identify snags in client-server communication.

Purpose	Closely monitors the RPC calls and promptly identifies snags in client-server communication		
Target of the test	NFS on Linux server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured.		
Outputs of the test	One set of results for every remotely mounted NFS		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

test	**Number of RPC calls:** Indicates the total number of RPC calls received from clients to the NFS server during the last measurement period.	Number	This is a good indicator of the workload on the server.
	Number of corrupted RPC requests: Indicates the total number of number of RPC calls with a length shorter than a minimum-sized RPC call during the last measure period.	Number	Ideally, the value of these measures should be 0. A non-zero value for these measures could indicate malformed NFS requests that can be caused by bugs in the client or server software or by physical network problems.
	Percentage of corrupted RPC requests: Indicates the percentage of truncated or damaged packets during the last measurement period.	Percent	

	Number of RPC call failures: Indicates the total number of calls rejected by the RPC layer in the NFS server during the last measurement period.	Number	Ideally, the value of these measures should be 0.
	Percentage of RPC call failures: Indicates the percentage of calls rejected by the RPC layer in the NFS during the last measurement period.	Percent	
	Number of bad authentication requests: Indicates the total number of bad authentication requests received from clients to the NFS server during the last measure period.	Number	The only time NFS performs authentication is when a client system attempts to mount the shared NFS resource. To limit access to the NFS service, TCP wrappers are used. TCP wrappers read the /etc/hosts.allow and /etc/hosts.deny files to determine if a particular client or network is permitted or denied access to the NFS service. Authentication errors can occur from bad /etc/hosts.allow entries. A high value for these measures is a cause for concern.
	Percentage of bad authentication requests: Indicates the percentage of bad authentication requests during the last measurement period.	Percent	

	Number of corrupted data headers: Indicates the number of RPC calls whose header could not be XDR decoded during the last measurement period.	Number	XDR is a standard for the description and encoding of data. It is useful for transferring data between different computer architectures, and it has been used to communicate data between diverse machines. All data in an RPC message is XDR encoded. The encoding of XDR data into transport buffers is referred to as "marshalling", and the decoding of XDR data contained within transport buffers and into destination RPC procedure result buffers, is referred to as "unmarshalling". Therefore, the process of marshalling takes place at the sender of any particular message, be it an RPC request or an RPC response. Unmarshalling, of course, takes place at the receiver. If 'unmarshalling' of an RPC request/response fails, it implies that the XDR decode has failed. Ideally, the value of this measure should be 0. A high value indicates too many malformed NFS requests, which can be caused by bugs in the client or server software or by physical network problems.
	Percentage of corrupted data headers: Indicates the percentage of corrupted data headers during the last measurement period.	Percent	

4.1.2 RPC Port Test

An NFS server listens on many TCP/UDP ports for RPC requests from clients. If too many RPC calls to the server fails, it could be because the TCP/UDP port configured for communication between the client and server is unavailable. This test periodically checks the NFS server for the availability of user-configured ports and promptly alerts administrators if one/more ports are found to be unavailable.

Purpose	Periodically checks the NFS server for the availability of user-configured ports and promptly alerts administrators if one/more ports are found to be unavailable
Target of the test	NFS on Linux server
Agent deploying the test	An internal agent

<table>
<tr><td><b>Configurable parameters for the test</b></td><td colspan="3">1. <b>TEST PERIOD</b> – How often should the test be executed
2. <b>Host</b> - The host for which the test is to be configured.
3. <b>rpc port</b> - Specify the TCP/UDP ports to be monitored. The specification should be in the following format: <i><ProtocolName>:<PortNo></i>. Multiple ports can be configured for monitoring as a comma-seperated list. For instance, <i>tcp:80,udp:90</i></td></tr>
<tr><td><b>Outputs of the test</b></td><td colspan="3">One set of results for every port specification</td></tr>
<tr><td rowspan="2"><b>Measurements made by the test</b></td><td><b>Measurement</b></td><td><b>Measurement Unit</b></td><td><b>Interpretation</b></td></tr>
<tr><td><b>Availability:</b>
Indicates whether this port is available or not.</td><td>Percent</td><td>The value 100 indicates availability and 0 indicates non-availability of the port.</td></tr>
</table>

Chapter

5

Monitoring NFS on Linux Clients

For monitoring NFS clients on Linux, the eG Enterprise Suite offers the *NFS Linux Client* monitoring model.

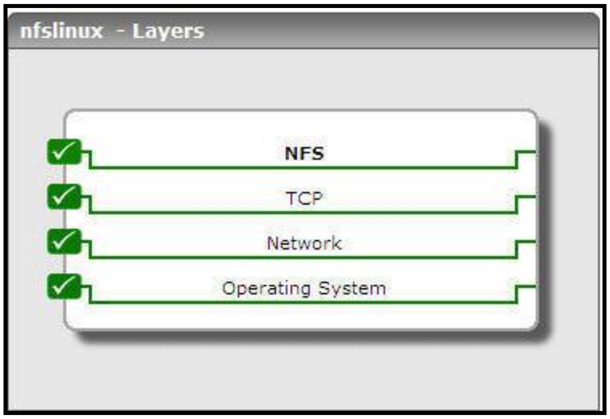


Figure 5.1: The NFS Linux Client monitoring model

Each layer of the model is mapped to tests that monitor the RPC calls made by the target NFS client to a server and promptly detect abnormalities in the RPC communication. Using the metrics so reported, admininistrators can find quick and easy answers for the following performance questions:

- Is the NFS client available over the network?
- Is the client utilizing the CPU, memory, and disk space resources optimally?
- Is the client overloading the server with too many RPC requests?
- Are too many RPC requests getting retransmitted to the server?
- Are the remote file systems available?
- Is the client taking too long to access the remote file systems?
- Is any NFS-mounted directory unavailable?
- Is any NFS-mounted directory consuming too much space?

Since the bottom 3 layers of Figure 5.1 have already been discussed extensively in the *Monitoring Unix and Windows* documents, let us focus on the first layer alone.

The sections that will follow discuss the first layer of the module alone.

5.1 The NFS Layer

The tests mapped to this layer monitor the following:

- Monitors NFS requests from clients and reports the number of retransmitted requests:
- Monitors the availability and access time of remote file systems
- Monitors the availability and space usage of each NFS-mounted directory

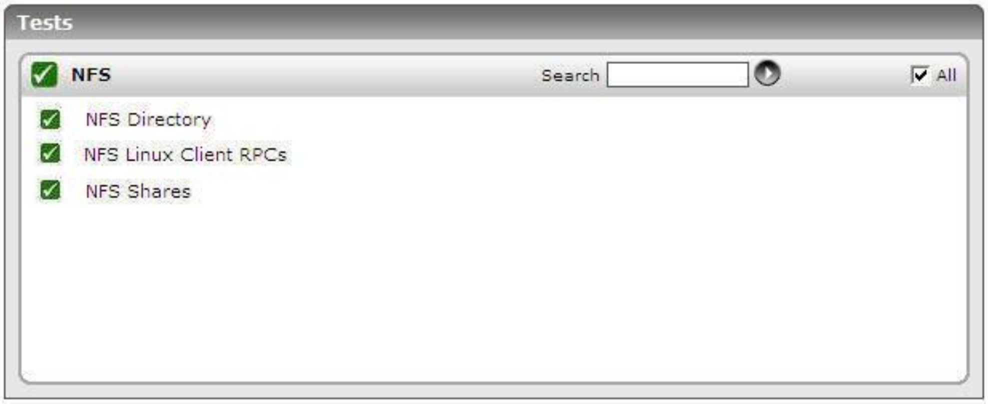


Figure 5.2: The tests mapped to the NFS layer

5.1.1 NFS Linux Client RPCs Test

This test monitors the NFS requests sent by clients, reports the total number of such requests, and reveals how many of these requests were retransmitted to the server. Retransmitted requests, if allowed to grow in number, can prove to be a serious bottleneck to the performance of the NFS. That way, this test, with its ability to promptly alert administrators to spikes in the number of retransmitted requests, is very useful.

Purpose	Monitors the NFS requests sent by clients, reports the total number of such requests, and reveals how many of these requests were retransmitted
Target of the test	NFS on Linux client
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured
Outputs of the test	One set of results for every remotely mounted NFS

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Number of RPC calls: Indicates the total number of RPC calls received from clients to the NFS server during the last measurement period.	Number	This is a good indicator of the workload on the server.

	Number of retransmitted requests: Indicates the total number of retransmitted RPC calls from clients during the last measurement period.	Number	Requests can be retransmitted due to dropped packets, socket buffer overflows, general server congestion, timeouts, etc. A high value for **No of retransmitted requests** and **Percentage of retransmitted requests** is hence, a cause for concern. One of the common reasons for request retransmission is the lack of sufficient number of NFS kernel threads on the server for processing client requests. The default number of threads for *rpc.nfsd* to start is typically eight threads. To tell *rpc.nfsd* to use more kernel threads, the number of threads must be passed as an argument to it. Typically, most distributions will have a file such as */etc/sysconfig/nfs* to configure this. In the file, increase this number — perhaps to 16 — on a moderately busy server, or increase up to 32 or 64 on a more heavily used system. Re-evaluate using *nfsstat* to determine whether or not the number of kernel threads is sufficient; if the retrans setting is 0 then it is enough; but, if the client still needs to retransmit, increase the number of threads further. Timeouts can also cause requests to be retransmitted. Two mount command options, **`timeo`** and **`retrans`**, control the behavior of UDP requests when encountering client timeouts due to dropped packets, network congestion, and so forth. The **`-o timeo`** option allows designation of the length of time, in tenths of seconds, that the client will wait until it decides it will not get a reply from the server, and must try to send the request again. The default value is 7 tenths of a second. The **`-o retrans`** option allows designation of the number of timeouts allowed before the client gives up, and displays the `Server not responding` message. The default value is 3 attempts. Once the client displays this message, it will continue to try to send the request, but only once before displaying the error message if another timeout occurs.

	Percentage of retransmitted requests: Indicates the retransmitted RPC calls from the clients during the last measurement period.	Percent	When the client reestablishes contact, it will fall back to using the correct **retrans** value, and will display the `Server OK` message. If you are already encountering excessive retransmission, If you are already encountering excessive retransmissions (see the output of the **nfsstat** command), or want to increase the block transfer size without encountering timeouts and retransmissions, you may want to adjust these values.
	Number of times authentication information had to be refreshed: Indicates the total number of times authentication information had to be refreshed on clients during the last measure period.	Number	

5.1.2 NFS Directory Test

This test reports statistics relating to NFS file systems remotely mounted by a client. This test auto discovers the remote file systems on a client and periodically accesses the file systems to check their availability and access times.

Purpose	Reports the availability and access time for NFS file systems remotely mounted by a client		
Target of the test	NFS on Linux Client		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **Host** - The host for which the test is to be configured.		
Outputs of the test	One set of results for every remotely mounted NFS		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Availability of the NFS file systems	Number	If the value of this measure is 0, it indicates that the file system is unavailable. The value 100 indicates the availability of the file system.

	Access time: Access time for the remotely mounted NFS file systems	Secs	By monitoring this value over time, an administrator can determine periods when NFS access is slow.

5.1.3 NFS Shares Test

Often, if an NFS file system fails, the directories mapped to the NFS file system will be unavailable. Accesses to these directories/files will take a long time and ultimately fail. This could potentially result in application failures and outages. Hence, administrators need the capability to detect when an NFS file system is unavailable or is running out of This test provides administrators with this capability.

This test executes on an NFS client, auto-discovers all NFS-mounted directories, and reports in real-time the availability and space usage of each of these directories.

Purpose	Reports in real-time the availability and space usage of NFS-mounted directory on an NFS client
Target of the test	An NFS Linux Client
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **Host** - The host for which the test is to be configured 3. **timeout** - Specify the maximum duration (in seconds) for which the test will wait for a response from the server. The default timeout period is 30 seconds. 4. **exclude file systems** – Provide a comma-separated list of file systems to be excluded from monitoring. By default, this is set to *none*, indicating that all file systems will be monitored by default. 5. **report by file system** – This test reports a set of measures for every NFS-mounted directory auto-discovered on a target NFS client – this implies that the discovered directory names will appear as descriptors of this test in the eG monitoring console. By selecting an option from the **report by file system** list, you can indicate how you want to display these directory names in the eG monitoring console. By default, the **Remote Filesystem** option is chosen; this indicates that, by default, the eG monitoring console will refer to each directory using the complete path to that directory in the remote file system – typically, this would include the name of the remote file system. For instance, if the *shares* directory on a remote host with IP *192.168.10.1* is being monitored, then the corresponding descriptor will be: *//192.168.10.1/shares*. If you choose the **Local Filesystem** option instead, then, the eG monitoring console will display only the name of the local file that is mapped to the remote directory – for example, if the *//192.168.10.1/shares* directory is locally mapped to the file */mnt*, then the descriptor will be */mnt*. Alternatively, you can have both the remote file system path and the local file mapping displayed in the eG monitoring console, by selecting the **Both** option from this list. In such a case, the descriptor will be of the format: *//192.168.10.1/shares (/mnt)*.		
Outputs of the test	One set of results for every NFS-mounted directory auto-discovered		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Indicates whether the directory is accessible or not.	Percent	The value 100 indicates that the mounted NFS is accessible. The value 0 indicates that the mounted NFS iss not accessible.
	Total capacity: Indicates the current total capacity of the mounted system disk partition.	MB	
	Used space: Indicates the amount of space currently used in a mounted system disk partition.	MB	

	Free space: Indicates the free space currently available on a disk partition of a mounted system.	MB	
	Percent usage: Indicates the percentage of space used on a mounted system disk partition.	Percent	Ideally, this value should be low. A high value or a value close to 100% is indicative of excessive space usage on this mounted system disk partition. If a number of NFS directories are exhibiting similar usage patterns, it is a definite cause for concern, as it indicates that the NFS file system as a whole coule be running out of space. If this situation is not brought under control soon, application failures and outages will become inevitable!

Chapter

6

Conclusion

This document has described in detail the monitoring paradigm used and the measurement capabilities of the eG Enterprise suite of products with respect to **Network File Systems**. For details of how to administer and use the eG Enterprise suite of products, refer to the user manuals.

We will be adding new measurement capabilities into the future versions of the eG Enterprise suite. If you can identify new capabilities that you would like us to incorporate in the eG Enterprise suite of products, please contact support@eginnovations.com. We look forward to your support and cooperation. Any feedback regarding this manual or any other aspects of the eG Enterprise suite can be forwarded to feedback@eginnovations.com.