
# Monitoring Distributed File Systems (DFS)

## eG Enterprise v6.0

**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 2003, and Windows 2000 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

# 1

# Monitoring Distributed File Systems

With Distributed File System (DFS), administrators can make it easy for users to access and manage files that are physically distributed across a network. They can make files distributed across multiple servers appear to users as if they reside in one place on the network. Users no longer need to know and specify the actual physical location of files in order to access them.

For example, if marketing material is scattered across multiple servers in a domain, administrators can use DFS to make it appear as though all of the material resides on a single server. This eliminates the need for users to go to multiple locations on the network to find the information they need.

Using DFS, administrators can group shared folders located on different servers by transparently connecting them to one or more DFS namespaces. Using the DFS tools, an administrator selects which shared folders to present in the namespace, designs the hierarchy in which those folders appear, and determines the names that the shared folders show in the namespace. When a user views the namespace, the folders appear to reside on a single, high-capacity hard disk. Users can navigate the namespace without needing to know the server names or shared folders hosting the data.

Moreover, DFS also offers the DFS Replication Service, with the help of which multiple copies of the same data can be created and stored in different namespace servers. This facilitates fault-tolerance and load-sharing.

It is hence evident that continuous availability, rapid and reliable access, and easy management of shared files and folders are the cornerstones of the DFS architecture. If DFS fails to deliver on these promises, users will be unable to access the files they want when they want it. This in turn will adversely impact user productivity and user confidence in the technology. This is why, administrators need to periodically run health checks on DFS and ensure that the DFS-managed files/folders are available and accessible at all times. To help administrators achieve this, eG Enterprise provides a specialized *Microsoft DFS* monitoring model.

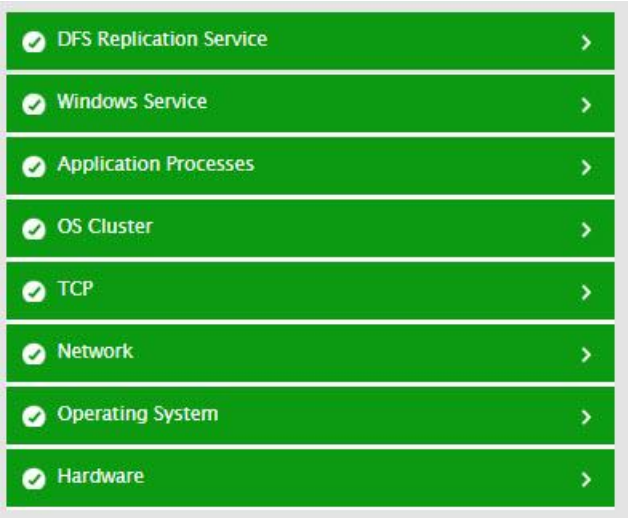


Figure 1.1: Layer model of the Microsoft DFS

Each layer of this model is mapped to tests that check the following at configured intervals:

- Are referral requests being processed quickly? Which DFS namespace is the slowest in responding to referral requests?
- How effective is the compression algorithm used by the DFS replication service? Is it saving bandwidth when performing replication? When replicating which folder was bandwidth saving the lowest? Are any replication connections consuming bandwidth excessively?
- Are the staging and Conflict and Deleted folders sized adequately for all replication folders?
- Is the quota configuration of the staging and conflict and deleted folders right?
- Is any replication folder experiencing replication bottlenecks? Which one?
- How is the API request load on the namespace server? Is the server able to handle the load?

These metrics shed light on the following:

- A potential slowdown when accessing a namespace on the namespace server;
- Sizing inadequacies of the namespace server
- Bottlenecks in replication and the replication folders they affect;
- Impact of replication on bandwidth usage;
- How quota configurations affect the speed and efficiency of replication;

The sections that follow elaborate on the **DFS Replication Service** layer alone, as the other layers have already been discussed in the *Monitoring Unix and Windows Servers* document.

# 1.1 The DFS Replication Service Layer

The tests mapped to this layer monitor the DFS replication service and reports abnormalities (if any).

## 1.1.1 DFS Namespace Referrals Test

A **DFS Namespace** is a virtual view of shared folders on different servers as provided by DFS. A DFS namespace consists of a root and many links and targets. The namespace starts with a root that maps to one or more root targets. Below the root are links that map to their own targets.

A **Referral** is a list of targets, transparent to the user, which a DFS client receives from DFS when the user is accessing a root or a link in the DFS namespace.

User experience with DFS is often measured by how quickly DFS processes referral requests. Prolonged delays in processing combined with frequent request failures can leave users frustrated. To prevent this, administrators must track the referral requests to each namespace managed by DFS, keep an eye out for request failures, monitor how DFS handles these requests, and promptly capture abnormalities. This is exactly what the **DFS Namespace Referrals** test does. For each namespace, this test reports the count of referral requests to the namespace, the percentage of requests that failed, and how swiftly the requests were processed. Besides proactively alerting administrators to processing bottlenecks, the test also leads them to the exact namespace where the problem lies.

| | |
|---|---|
| **Purpose** | For each namespace, this test reports the count of referral requests to the namespace, the percentage of requests that failed, and how swiftly the requests were processed |
| **Target of the test** | A server that hosts the DFS namespace (this can even be a server that contains the Dfs root or a replica of it) |
| **Agent deploying the test** | An internal agent |
| **Configurable parameters for the test** | 1. **TEST PERIOD** - How often should the test be executed<br>2. **HOST -** The host for which the test is to be configured.<br>3. **PORT** – The port at which the **HOST** listens. |

| | | | |
|---|---|---|---|
| **Outputs of the test** | One set of results for each DFS namespace to which the namespace server being monitored belongs | | |
| **Measurements made by the test** | **Measurement** | **Measurement Unit** | **Interpretation** |
| | **Avg response time:**<br>Indicates the average time taken by the DFS namespace service to respond to referral requests to this namespace during the last measurement period. | Secs | Ideally, the value of this measure should be low. A consistent increase in the value of this measure is indicative of poor responsiveness of the DFS namespace service.<br>Compare the value of this measure across namespaces to know which namespace is responding slowly to referral requests. |
| | **Requests processed**:<br>Indicates the number of referral requests processed by the DFS namespace service for this namespace during the last measurement period. | Number | |
| | **Requests failed:**<br>Indicates the number of requests for this namespace that failed. | Number | Ideally, the value of this measure should be 0. |
| | **Percentage of failed requests:**<br>Indicates the percentage of referral requests to this namespace that failed. | Percent | The value 0 is desired for this measure. A high value is a cause for concern. Compare the value of this measure across namespaces to know requests to which namespace are failing often. |
| | **Request processing rate:**<br>Indicates the rate at which the referral requests to this namespace are processed by the DFS namespace service. | Requests/Sec | A high value is desired for this measure. A steady dip in this value is indicative of a processing bottleneck. Compare the value of this measure across namespaces to know the requests that are being serviced very slowly. |

## 1.1.2 DFS Replication Folders Test

DFS Replication is an efficient, multiple-master replication engine that you can use to keep folders synchronized between servers across limited bandwidth network connections.

DFS Replication uses a compression algorithm known as remote differential compression (RDC). RDC detects changes to the data in a file and enables DFS Replication to replicate only the changed file blocks instead of the entire file.

To use DFS Replication, you must create replication groups and add replicated folders to the groups. Replication groups, replicated folders, and members are illustrated in the following figure.

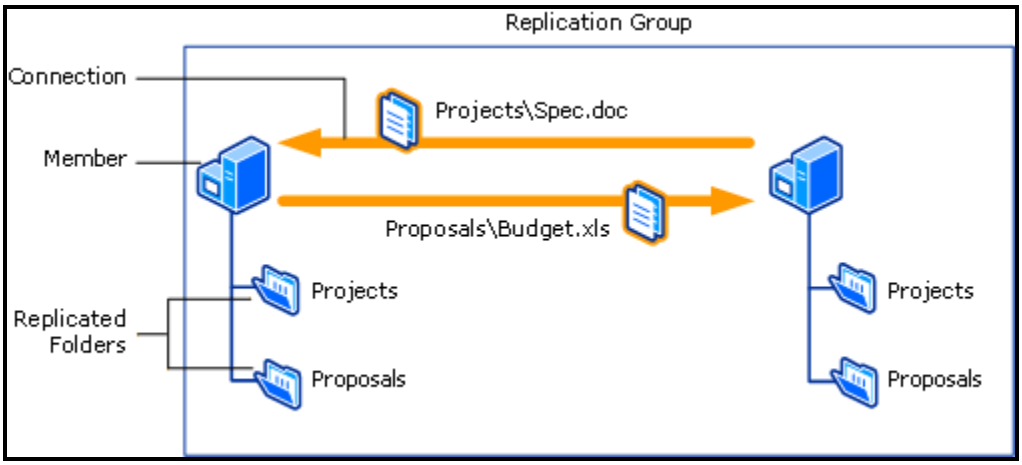


Figure 1: How DFS replication works

This figure shows that a *replication group* is a set of servers, known as *members* , which participates in the replication of one or more replicated folders. A *replicated folder* is a folder that stays synchronized on each member. In the figure, there are two replicated folders: Projects and Proposals. As the data changes in each replicated folder, the changes are replicated across connections between the members of the replication group. The connections between all members form the replication topology.

DFS Replication uses staging folders to act as caches for new and changed files to be replicated from sending members to receiving members. The sending member begins staging a file when it receives a request from the receiving member. The process involves reading the file from the replicated folder and building a compressed representation of the file in the staging folder. This is the *staged file*. After being constructed, the staged file is sent to the receiving member; if remote differential compression [RDC] is used, only a fraction of the staging file might be replicated. The receiving member downloads the data and builds the file in its staging folder. After the file has completed downloading on the receiving member, DFS Replication decompresses the file and installs it into the replicated folder. Each replicated folder has its own staging folder, which by default is located under the local path of the replicated folder in the DfsrPrivate\Staging folder.

DFS Replication uses a "last-writer wins" method for determining which version of a file to keep when a file is modified on two or more members. The losing file is stored in the Conflict and Deleted folder on the member that resolves the conflict. This member might not be the member where the changes originated. Each replicated folder has its own Conflict and Deleted folder, which is located under the local path of the replicated folder in the DfsrPrivate\ConflictandDeleted folder.

Slow replication can cause replicated folders to remain out-of-sync across members for long time periods. Because of this, users may end up receiving an inconsistent/incomplete view of data. Typically, replication may slow down owing to the following:

- The lack of adequate bandwidth resources;
- Staging and conflict and deleted folders that have not been sized to deal with high levels of replication activity;

By closely monitoring these factors, administrators can detect bottlenecks to replication early and take pre-emptive measures. This is where the **DFS Replication Folders** test helps. This test auto-discovers replicated folders and for each folder reports the bandwith saved during replication and tracks the growth in size of the associated staging folders and config and deleted items folders. This way, administrators can proactively detect probable slowdowns in replication. In addition, they can pinpoint what is causing the slowdown and which replication folders will be affected

by this.

| | | | |
|---|---|---|---|
| **Purpose** | Auto-discovers replicated folders and for each folder reports the bandwith saved during replication and tracks the growth in size of the associated staging folders and config and deleted items folders. This way, administrators can proactively detect probable slowdowns in replication. In addition, they can pinpoint what is causing the slowdown and which replication folders will be affected by this. | | |
| **Target of the test** | A namespace server that hosts a replicated folder | | |
| **Agent deploying the test** | An internal agent | | |
| **Configurable parameters for the test** | 1. **TEST PERIOD** - How often should the test be executed<br>2. **HOST -** The host for which the test is to be configured.<br>3. **PORT** – The port at which the **HOST** listens. | | |
| **Outputs of the test** | One set of results for each replicated folder | | |
| **Measurements made by the test** | **Measurement** | **Measurement Unit** | **Interpretation** |
| | **Bandwidth savings using DFS replication:**<br>Indicates the percentage of bandwidth that was saved by the DFS Replication service for the replicated folder.<br>. | Percent | A high degree of compression results in high bandwidth savings and faster/inexpensive replication. A high value is therefore desired for this measure. A low value is often an outcome of a poor compression algorithm. |
| | **Conflict files generated:**<br>Indicates the number of files and folders in this replicated folder that were moved to the conflict and deleted folder during the last measurement period. | Number | The value of this measure is a good indicator of how often the files in a replication folder are changed. |

| | | | |
|---|---|---|---|
| | **Conflict files cleanups**:<br>Indicates the number of conflict loser files and folders that were deleted from the conflict and deleted folder of this replicated folder during the last measurement period. | Number | The Conflict and Deleted folder has high and low watermarks (90 percent and 60 percent of Conflict and Deleted folder quota, respectively) that govern cleanup and excessive usage of the folder.<br>This implies that if the size of the Conflict and Deleted folder is over 90% of the quota configured for it, DFS will start deleting files from the folder automatically and will keep at it until the size of the folder touches its low watermark – i.e., until size becomes 60% of the configured quota.<br>If the value of this measure is consistently high for a replicated folder, it can only mean that the Conflict and Deleted folder is continuously growing in size, thus forcing DFS to cleanup old files from it on a regular basis.<br>This could also indicate that files in the corresponding replicated folder are changing very often, thus adding to the contents and the size of the Conflict and Deleted folder. Another reason for a large number of cleanups could be a low quota configured for the Conflict and Deleted folder. |
| | **Deleted files generated:**<br>Indicates the number of replicated deleted files and folders that were moved from this replicated folder to the conflict and deleted folder during the last measurement period. | Number | The Conflict and Deleted folder can also be used to store files that are deleted from replicated folders. |

| | | | |
|---|---|---|---|
| | **Deleted files cleanups:**<br>Indicates the number of replicated deleted files and folders that were deleted from the conflict and deleted folder of this replicated folder during the last measurement period.<br>. | Number | Deleted files are treated like conflict files in that they are purged when the Conflict and Deleted folder reaches 90 percent of the configured quota.<br>If the size of the Conflict and Deleted folder grows above 90% of its configured quota, DFS will start deleting files from the folder automatically and will keep at it until the size of the folder touches its low watermark – i.e., until size becomes 60% of the configured quota.<br>If the value of this measure is consistently high for a replicated folder, it can only mean that the Conflict and Deleted folder is continuously growing in size, thus forcing DFS to cleanup old files from it on a regular basis.<br>This could also indicate that files in the corresponding replicated folder are either changing or are being deleted very often, thus adding to the contents and the size of the Conflict and Deleted folder. |
| | **Staging files generated:**<br>Indicates the number of times files in this replicated folder were staged during the last measurement period. | Number | The value of this measure is a good indicator of the number of times files were added to or changed in a replication folder.<br>A high value is therefore indicative of a high level of replication activity. A consistent and significant increase in the value of this measure over time could also indicate that staging files are not being sent to the receiving member as quickly as they are created by the sending member – this in turn could indicate a bottleneck in replication. |

| | | | |
|---|---|---|---|
| | **Staging files cleanups:**<br>Indicates the number of files and folders that were cleaned up from the staging folder of this replication folder by the DFS Replication service during the last measurement period. | Number | The DFS Replication service stages files and folders in the staging folder before they are replicated, and automatically cleans up the staging folder when it exceeds a pre-configured threshold of the quota.<br>If the value of this measure remains high consistently, it could indicate that the staging folder is rapidly exhausting its configured quota size, causing DFS to keep deleting old staging files from the folder. While this can happen if the level of replication activity is high, it can also happen if the staging folder is configured with a low quota size. If a staging folder quota is configured to be too small, DFS Replication might consume additional CPU and disk resources to regenerate the staged files. Replication might also slow down because the lack of staging space can effectively limit the number of concurrent transfers with partners. |
| | **Conflict space in use:**<br>Indicates the total size of conflict loser files and folders in the conflict and deleted folder of this replicated folder during the last measurement period. | MB | The quota size of the Conflict and Deleted folder is 660 MB by default; however, this can be changed. Also, the Conflict and Deleted folder is configured with a high watermark of 90% and a low watermark of 60% of the quota. If the value of this measure is over 90% of the default/configured (as the case may be) quota size, then DFS will start purging files from the conflict and deleted folder until its size falls to 60% of the quota.<br>Compare the value of this measure across replication folders to know which replication folder's conflict and deleted folder is growing close to its configured quota size. |
| | **Deleted space in use:**<br>Indicates the total size of replicated deleted files and folders currently in the conflict and deleted folder of this replicated folder. | MB | Deleted files are treated like conflict files in that they are purged when the Conflict and Deleted folder reaches 90 percent of the configured quota. |

| | | | |
|---|---|---|---|
| | **Staging space in use:**<br>Indicates the total size of the files and folders in the staging folder of this replication folder during the last measurement period. | MB | The default size of each staging folder is 4,096 MB. This is not a hard limit, however. It is only a quota that is used to govern cleanup and excessive usage based on high and low watermarks (90 percent and 60 percent of staging folder size, respectively). This quota can be changed.<br>If the value of this measure reaches 90% of the configured quota, the oldest staged files are purged until the staging folder reaches 60 percent of the configured quota.<br>For good operational performance, increasing the quota size of a staging folder is recommended when you have multiple large files that change frequently. Microsoft also recommends that you increase the staging folder quota on hub members that have many replication partners.<br>If a staging folder quota is configured to be too small, DFS Replication might consume additional CPU and disk resources to regenerate the staged files. Replication might also slow down because the lack of staging space can effectively limit the number of concurrent transfers with partners. |

### 1.1.3 DFS Namespace API Queue Test

To manage DFS, administrators often rely on management tools and functions that Windows offers – for eg., DFS Management UI, DFS BPA, DFS Namespaces Windows PowerShell cmdlets, File Server Resource Manager, and many more. All these management tools use NetDfs API functions to perform a wide variety of management tasks such as:

- Adding a DFS link to a DFS root;
- Creating or removing stand-alone and domain-based DFS namespaces;
- Adding targets to an existing DFS link;
- Removing a DFS link from a DFS root;
- Removing a target from a DFS link;
- Viewing and configuring information about DFS roots and links

In the process, these tools can generate a significant amount of management traffic on the namespace server. The true test of strength of the namespace server lies in how well it processes this traffic. To determine whether/not the namespace server is able to handle this traffic, administrators can use the **DFS Namespace API Queues** test. This test continuously tracks the length of the API queues on the server to figure out how quickly the server processes the API requests in the queue. This sheds light on the load imposed by API requests on the server and the ability of the server to respond to these requests.

| | | | |
|---|---|---|---|
| **Purpose** | Continuously tracks the length of the API queues on the server to figure out how quickly the server processes the API requests in the queue | | |
| **Target of the test** | A DFS namespace server | | |
| **Agent deploying the test** | An internal agent | | |
| **Configurable parameters for the test** | 1. **TEST PERIOD** - How often should the test be executed<br>2. **HOST -** The host for which the test is to be configured.<br>3. **PORT** – The port at which the **HOST** listens. | | |
| **Outputs of the test** | One set of results for the DFS namespace server being monitored | | |
| **Measurements made by the test** | **Measurement** | **Measurement Unit** | **Interpretation** |
| | **API queue length:**<br>Indicates the number of requests (made using the NetDfs API) currently in the queue for the DFS Namespace service to process. | Number | If the value of this measure keeps increasing while at the peak level of activity, it could indicate a bottleneck in processing API requests. One of the common reasons why a namespace server may be unable to process API requests quickly is improper server sizing.<br>A server that is sized right should be able to crank through its work queues and be responsive. A server with insufficient resources on the other hand will not be able to handle this load, and may hence end-up processing requests slowly; this in turn will increase the length of the API request processing queues.<br>In such cases, add more processing power to the server and see if it helps reduce queue length. |

### 1.1.4 DFS Replication Connections Test

The replication topology consists of the logical connections that DFS Replication uses to replicate files among servers. Members in a replication topology communicate via two one-way connections. These two connections allow data to flow in both directions.

Where replication activities consume considerable bandwidth, administrators may want to quickly identify the bandwidth-intensive replication connections, determine how much bandwidth is currently saved by these connections by compressing the traffic, and decipher how the compression algorithm can be tweaked to conserve more bandwidth. This is exactly what the **DFS Replication Connections** test does. This test monitors each replication connection to a namespace server and reports the throughput and bandwidth savings of each connection, so that administrators can identify those connections that could be candidates for additional traffic compression.

| | | | |
|---|---|---|---|
| **Purpose** | Monitors each replication connection to a namespace server and reports the throughput and bandwidth savings of each connection, so that administrators can identify those connections that could be candidates for additional traffic compression | | |
| **Target of the test** | A DFS namespace server | | |
| **Agent deploying the test** | An internal agent | | |
| **Configurable parameters for the test** | 1. **TEST PERIOD** - How often should the test be executed<br>2. **HOST -** The host for which the test is to be configured.<br>3. **PORT** – The port at which the **HOST** listens. | | |
| **Outputs of the test** | One set of results for each replication connection to the namespace server monitored | | |
| **Measurements made by the test** | **Measurement** | **Measurement Unit** | **Interpretation** |
| | **Total files received:**<br>Indicates the number of files received on this connection during the last measurement period. | Number | Compare the value of this measure across connections to know which connection receives the maximum number of files. |
| | **Total data received:**<br>Indicates the total number of bytes received on this connection during the last measurement period. | MB | Compare the value of this measure across connections to know which connection receives the maximum data. |
| | **RDC data received:**<br>Indicates the bytes that were received on this connection while replicating files using remote differential compression during the last measurement period. | MB | This is the actual bytes received over the network without the networking protocol overhead.<br>By comparing the value of this measure with that of the *Total data received* measure for a connection, administrators can determine whether/not data compression has improved throughtput and reduced bandwidth consumption of that connection. |

| | | | |
|---|---|---|---|
| | **Bandwidth savings using DFS replication:**<br>Indicates the percentage of bandwidth that was saved by the DFS Replication service. | Percent | This measure indicates how much bandwidth was saved for this connection using a combination of remote differential compression (RDC) and other compression technologies that minimize network bandwidth use. For example, a value of 20 indicates that the DFS Replication service used 20% less bandwidth than it would have used if it had transmitted the entire files uncompressed over the network.<br>By comparing the value of this measure across connections, administrators can identify that connection which has saved the least bandwidth. For such connections, the compression technologies employed may have to be tweaked further to ensure optimal bandwidth usage. |

## 1.1.5 DFS Namespace API Requests Test

The management tools (eg., DFS Management UI, DFS BPA, DFS Namespaces Windows PowerShell cmdlets, File Server Resource Manager) used by administrators to manage DFS typically employ NetDfs API functions to perform critical management tasks – eg., creating/removing namespaces, adding roots and links, viewing/configuring information about roots and links, and so on. In the process, these tools impose a significant amount of operational load on the namespace server. If the server is not sized right to handle this load, API requests may either fail frequently or may end-up being processed slowly by the server, thus increasing the DFS management challenges of administrators. This in turn will impact the user experience with DFS. If such situations are to be avoided, administrators should use the **DFS Namespace API Requests** test to track the number and type of API requests to the namespace server, measure how quickly the server processes requests of each type, identify request types that failed often, and accordingly, decide whether/not the server should be resized to improve its processing ability.

| | |
|---|---|
| **Purpose** | Tracks the number and type of API requests to the namespace server, measures how quickly the server processes requests of each type, pinpoints request types that were most often rejected by the server, and accordingly, helps administrators decide whether/not the server should be resized to improve its processing ability |
| **Target of the test** | A server that hosts the DFS namespace (this can even be a server that contains the Dfs root or a replica of it) |
| **Agent deploying the test** | An internal agent |
| **Configurable parameters for the test** | 1. **TEST PERIOD** - How often should the test be executed<br>2. **HOST -** The host for which the test is to be configured.<br>3. **PORT** – The port at which the **HOST** listens. |
| **Outputs of the test** | One set of results for each type of API request to the namespace server being monitored |

| Measurements made by the test | Measurement | Measurement Unit | Interpretation |
|---|---|---|---|
| | **Avg response time:**<br>Indicates the average time taken by the DFS namespace server to respond to API requests of this type. | Secs | Ideally, the value of this measure should be low. A consistent increase in the value of this measure is indicative of poor responsiveness of the DFS namespace service.<br>Compare the value of this measure across request types to know which API requests are being processed slowly by the server. |
| | **Requests processed**:<br>Indicates the number of API requests of this type processed by the server since the last measurement period. | Number | |
| | **Requests failed:**<br>Indicates the number of API requests of this type that failed during the last measurement period. | Number | Ideally, the value of this measure should be 0. |
| | **Percentage of failed requests:**<br>Indicates the percentage of API requests of this type that failed. | Percent | The value 0 is desired for this measure. A high value is a cause for concern. Compare the value of this measure across types to know which type of requests is prone to failures. |
| | **Request processing rate:**<br>Indicates the rate at which the API requests of this type are processed. | Requests/Sec | A high value is desired for this measure. A steady dip in this value is indicative of a processing bottleneck. Compare the value of this measure across types to know the requests that are being serviced very slowly. |

### 1.1.6 DFS Replication Volumes Test

In DFS, replication folders are hosted on volumes. The DFS replication service maintains one ESE (Extensible Storage Engine) database per volume. This database is used to store metadata about each file and folder in the replicated folder.

How quickly the DFSR performs replication depends upon how well the volume (i.e., the ESE database in the volume) is sized. Because volumes are to be sized on the basis of the transaction load that replication imposes on them, administrators may first want to ascertain the level of database activity that replication typically generates per volume and then plan volume capacity accordingly. The **DFS Replication Volumes** test helps administrators with this exercise.

For every volume that hosts replication folders, this test reports the count of database lookups and commits performed on the volume since the last measurement period. By tracking variations to these measures over time, administrators can easily understand the current load, forecast the future load, and use these results to determine/fine-tune the volume size.

<table>
<tr><td>Purpose</td><td colspan="3">For every volume that hosts replication folders, this test reports the count of database lookups and commits performed by the volume since the last measurement period.</td></tr>
<tr><td>Target of the test</td><td colspan="3">A server that hosts the DFS namespace (this can even be a server that contains the Dfs root or a replica of it)</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed<br>2. HOST - The host for which the test is to be configured.<br>3. PORT – The port at which the HOST listens.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each volume hosting replication folders</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Database lookups:<br>Indicates the number of database search operations performed by the DFS replication service on this volume since the last measurement period.</td><td>Number</td><td rowspan="2">These measures are good indicators of the level of database activity that replication generates on a volume. Compare the value of these measures across volumes to know which volume is used the most. For this volume, closely track variations in the value of these measures over time, understand load changes, and accordingly right-size (if required) the volume.</td></tr>
<tr><td>Database commits:<br>Indicates the number of database commit operations performed by the DFS replication service on this volume since the last measurement period.</td><td>Number</td></tr>
</table>

2

# 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 **the Distributed File System (DFS)**. 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.