
Using SYMCLI and GNS to Propagate Group Definitions to Multiple Hosts

Abstract

This white paper describes using Group Name Services (GNS) in an open systems environment to propagate SYMAPI device group and composite group definitions across multiple hosts and Symmetrix arrays.

Published 1/25/2005

Copyright © 2005 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS”. EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, Symmetrix, Celerra, CLARiiON, CLARAlert, Documentum, HighRoad, Legato, Navisphere, PowerPath, ResourcePak, SnapView/IP, SRDF, TimeFinder, VisualSAN, and where information lives are registered trademarks and EMC Automated Networked Storage, EMC ControlCenter, EMC Developers Program, EMC OnCourse, EMC Proven, EMC Snap, Access Logix, AutoAdvice, Automated Resource Manager, AutoSwap, AVALONidm, C-Clip, Celerra Replicator, Centera, CentraStar, CLARevent, Connectrix, CopyCross, CopyPoint, DatabaseXtender, Direct Matrix, Direct Matrix Architecture, EDM, E-Lab, Enginuity, FarPoint, FLARE, GeoSpan, InfoMover, MirrorView, NetWin, OnAlert, OpenScale, Powerlink, PowerVolume, RepliCare, SafeLine, SAN Architect, SAN Copy, SAN Manager, SDMS, SnapSure, SnapView, StorageScope, SupportMate, SymmAPI, SymmEnabler, Symmetrix DMX, Universal Data Tone, and VisualSRM are trademarks of EMC Corporation. All other trademarks used herein are the property of their respective owners.

Part Number 300-001-384 REV C

Table of Contents

Introduction	4
Purpose and Scope	4
Related Documentation	4
Practical Uses	4
Using Group Name Services (GNS).....	5
GNS Behavior in Client/Server Mode	6
The GNS Daemon.....	7
Starting the GNS Daemon	7
Using the GNS Daemon Options File	8
Using a Command to Revise Configuration Settings in the GNS Daemon Options File.....	9
Implementing User Authorization for the GNS Daemon	9
Mirroring an RDF Group Definition on a Remote Symmetrix Array	10
Maintaining a Consistency Support Database	11
Restrictions	11
Resolving Duplicate Group Names	12
Backing Up and Restoring GNS Data.....	12
Diagnostics	13
Example 1: Importing Device Groups into GNS	14
Example 2: Using GNS to Propagate a Composite Group Definition.....	18

Introduction

This paper describes using Group Name Services (GNS) in an open systems environment¹ to propagate SYMAPI device group and composite group definitions across multiple hosts and Symmetrix[®] arrays. GNS stores device group and composite group definitions in a shared repository that is accessible from multiple hosts. If you make a change to a group definition on one host, that change is visible to other hosts. Thus, you can perform SYMCLI and SYMAPI control operations from any host in the Symmetrix environment.

A *device group* is a user-defined group of devices that belong to a *single* Symmetrix array and a single RDF (RA) group. In versions of EMC[®] Solutions Enabler prior to version 5.4, a device group definition was stored only in the SYMAPI database file on the host where you defined the group. To perform control operations from another host, you needed to copy the group definition to another host. Now you can use GNS to automatically propagate group definitions and their updates to the Symmetrix arrays and other locally attached hosts that are running the GNS daemon.

A *composite group* is a user-defined group of devices that can span *multiple* Symmetrix arrays. You can define an RDF composite group and enable it for RDF consistency protection, using PowerPath[®] or SRDF/A MSC support. If you store such a group within GNS, any change you make to the group (such as adding a device to it) is automatically added to the PowerPath or SRDF/A MSC database on all hosts that are running GNS and PowerPath, or GNS and the RDF daemon, respectively.

In an SRDF[®] environment, GNS provides an option to automatically mirror an RDF1 or RDF2 group to the remote Symmetrix array for activation and use during a disaster failover event.

Purpose and Scope

This document describes GNS functionality included in EMC Solutions Enabler up through version 6.0 and Enginuity[™] version 5x71. While this paper discusses the implementation of GNS on open systems platforms, similar functionality may exist on the mainframe platform (refer to the *OS/390 and z/OS Product Set* product guides).

Related Documentation

GNS works with various EMC open systems products, including the TimeFinder[®] product family and the SRDF product family. The following white papers provide information about these products:

- *Using SYMCLI to Perform Control Operations with SRDF Family Products* (P/N 300-000-076)
- *Using SYMCLI to Perform TimeFinder/Mirror Control Operations* (P/N 300-000-074)
- *Using SYMCLI to Perform TimeFinder/Snap Operations* (P/N 300-001-381)
- *Using SYMCLI to Perform TimeFinder/Clone Operations* (P/N 300-001-382)

Practical Uses

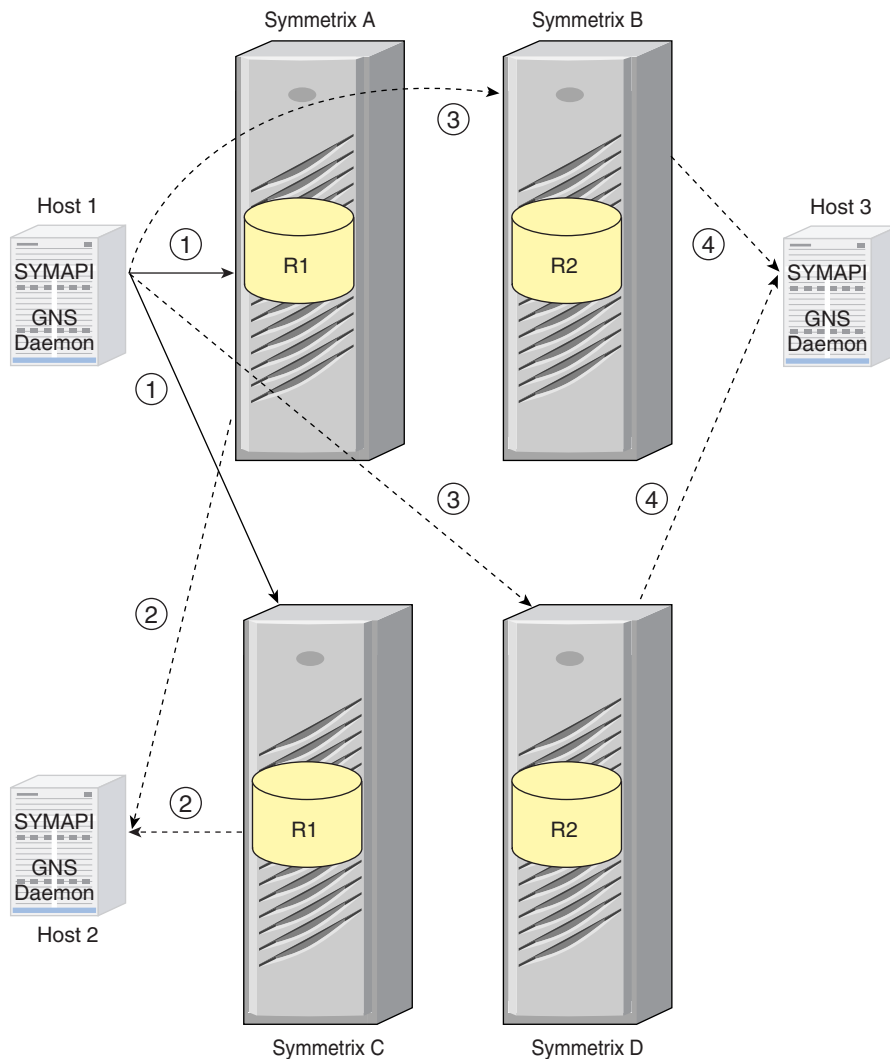
GNS allows you to maintain updated versions of group definitions on each control host in a Symmetrix environment. If you make a change to a group definition on one host, that change is visible quickly to other hosts. In the event that one host goes down, you can still perform SYMCLI control operations from another host in your Symmetrix environment. In an SRDF environment, GNS allows you to automatically mirror an

¹ Currently, GNS may maintain mainframe group definitions but independently of the open system group definitions. Open systems environments ignore mainframe groups stored within GNS. Likewise, mainframe environments ignore open system groups. For information about using GNS in a mainframe environment, refer to the mainframe documentation.

RDF1 device group or composite group to the remote Symmetrix array for activation and use if a failover to the R2 side is required or, conversely, if a failback to the R1 side is required.

Using Group Name Services (GNS)

If GNS is enabled, group definitions are stored in a shared global repository rather than in a SYMAPI database file that resides on an individual host. As Figure 1 illustrates, each GNS-enabled host sees the same group definitions as well as any updates to these definitions that are made by other hosts. The configuration represents two local Symmetrix arrays (A and C) and remote Symmetrix arrays B and D.



CLI-000082

Figure 1. Propagating Group Definition Changes Via Group Name Services (GNS)

If a Host-1 application in Figure 1 makes a change to a group definition that includes devices on Symmetrix A, the GNS daemon there propagates the updated group definition to Symmetrix A (step 1), which in turn propagates the update to Host 2 in step 2. Local Symmetrix C is not involved unless a composite group definition spans Symmetrix arrays A and C. Step 3 illustrates the group update being propagated to the remote Symmetrix arrays B and D. Step 4 shows that the Host-3 GNS daemon notices the updated group definition, although perhaps much later following a recovery action, and makes the propagated changes available to applications there.

For a composite group definition that spans multiple Symmetrix arrays such as Symmetrix arrays A and C, the GNS shared repository spans the same Symmetrix arrays. Each Symmetrix array contains that portion of the group definition relevant to it (the group's devices present on that array). If you define on Host 1 a composite group that includes devices on Symmetrix arrays A and C, that definition is propagated to locally connected host 2 via steps 1 and 2. Steps 3 and 4 illustrate the composite group update being propagated to the remote Symmetrix arrays and the remote host.

Setting the SYMAPI_USE_GNS option in the SYMAPI options file to ENABLE² enables GNS. When GNS is enabled on a host, group operations are performed through the common GNS infrastructure. When GNS is disabled, group operations are performed against the local SYMAPI database file in the same way as it was prior to GNS.

When GNS is enabled, group definitions are stored in the global GNS repository instead of in individual SYMAPI database files. While GNS is enabled, you have limited access to groups that existed in the SYMAPI database before GNS was enabled. You can import one or all of these groups into GNS.

You can use relevant `list` commands to display the inactive SYMAPI database groups on your host:

```
symdbg list -inactive
symcgc list -inactive
```

You can use `activate` commands to import a device group or composite group into GNS. For example, if the groups MyDG and MyCG are inactive SYMAPI database groups, the following commands import them into GNS:

```
symdbg activate MyDG
symcgc activate MyCG
```

To import *all* inactive SYMAPI database groups into GNS, use the `activateall` commands instead. For example:

```
symdbg activateall
symcgc activateall
```

Note that an `activate` or `activateall` command may fail if a group name that you are attempting to import is already in use within GNS.

Although you can issue the `activate` commands when GNS is *disabled*, it is recommended that you *not* do it. In this case, you would be accessing a copy of GNS group definitions that were written into the SYMAPI database file for diagnostic purposes when GNS was previously enabled.

Certain group attributes such as a gatekeeper list are specific to a host and are not stored within GNS itself. These attributes are held within the SYMAPI database file (along with the list of GNS groups) and are merged with a group definition that is read from GNS. Thus, if you were to delete a host's SYMAPI database file, you would delete these host-specific attributes.

GNS Behavior in Client/Server Mode

In client/server mode, client group operations are sent to the server, which may hand off these operations to GNS for servicing. In this case, the SYMAPI server acts as the GNS client, and the *server's* SYMAPI options file (SYMAPI_USE_GNS) determines whether GNS is enabled for the client. If GNS is enabled and the client is caching (SYMAPI_CONN_TYPE_REMOTE_CACHED), the client checks periodically with the server to determine if the group definitions that it is caching need to be updated.

² Setting this option to ENABLE enables GNS for any applications that are using the default SYMAPI database file. Typically, applications that run with a private database file intend for their modifications to remain private, and therefore do not make use of GNS. The value DISABLE disables GNS on the host.

The SYMAPI_GNS_CS_STALE_DATA_TIMEOUT option in the *client's* SYMAPI options file controls how frequently the client checks the server. The default is one second. When a client needs group information, it validates its cached group definitions with the server if the time since the last check is longer than the default value or whatever new value you have set for this option.

The GNS Daemon

The GNS daemon is the only entity that directly accesses the GNS repository. GNS client requests (for example, SYMCLI and SYMAPI requests) are forwarded to the GNS daemon, and the daemon performs them.

Each host performing management operations runs an instance of the GNS daemon. In client/server mode, the GNS daemon runs on the server host.

The GNS daemon monitors GNS repositories on all directly-connected Symmetrix arrays for changes made by other GNS daemons on other hosts. Such changes are reflected in the daemon's in-memory cache and made available to local clients. As mentioned previously, changes made to an RDF group cause the daemon to automatically reflect those changes in a remote mirror copy of the group (provided that you set the `gns_remote_mirror` option to ENABLE).

Starting the GNS Daemon

There are three ways to start the GNS daemon:

- Automatically if GNS is enabled.

This will occur the first time the Solutions Enabler run-time libraries attempt to connect with the daemon (that is, the first time you try doing something like `symdg list` that requires GNS). However, prestarting the daemon using the following methods is frequently preferable as the daemon may take some time to initially construct its cache. The time required for this task depends on how many groups and Symmetrix arrays the daemon has to load.

- Manually. Use the `stordaeon` command line utility and the following command:

```
stordaeon start storgnsd
```

- Automatically each time the local host is booted. Use the `stordaeon` command line utility and the following command:

```
stordaeon install storgnsd -autostart
```

Automatic start at host boot and a Solutions Enabler “watchdog” mechanism combine to automatically restart a running daemon if it crashes. The watchdog mechanism (enabled by default) is controlled by the autorestart entry in the daemon options file (refer to the section “Using the GNS Daemon Options File”). This combination of the watchdog and automatic start at host reboot guarantees that the daemon always runs unless there are extenuating circumstances. The watchdog attempts a maximum of three restarts within a 15-minute period. If there is a third crash within this 15-minute period, no restart is attempted.

For information about stopping the GNS daemon (rarely required or desirable), querying daemon status, or querying the daemon's log files, refer to the *EMC Solutions Enabler Symmetrix Base Management CLI Product Guide*.

Using the GNS Daemon Options File

You can set various GNS configuration options within the common daemon options file. The location of this file depends upon your platform:

- UNIX: /var/symapi/config/daemon_options
- Windows: c:\Program Files\EMC\SYMAPI\config\daemon_options

You specify options using the syntax `storgnsd:OptionName=OptionValue`. For example:

```
storgnsd:gns_remote_mirror=enable
storgnsd:gns_symavoid=000000001111,000000002222
```

Table 1 describes some of the more useful GNS configuration options.

Table 1. Options Available for the GNS Daemon

Option	Description
autorestart	Makes use of the watchdog mechanism, which automatically restarts a running daemon if it crashes. The default is ENABLE. To cancel this behavior, set the option to DISABLE.
gns_device_poll_interval	The interval in seconds that you want to use to poll locally attached Symmetrix arrays for group changes. The default is 15 seconds.
gns_device_remote_poll_interval	Interval in seconds that you want to use to poll remotely attached Symmetrix arrays to see if pending changes can be propagated there. The default is 60 seconds.
gns_rmtarr_update_interval	Interval in seconds to use to check for group changes that need to be propagated to remote Symmetrix arrays. The default is 60 seconds.
gns_symdiscover_interval	Interval in seconds to use to check whether there are new Symmetrix arrays that need to be interrogated. The default is 600 seconds.
gns_ppath_poll_interval	Interval in seconds to use to check whether PowerPath definitions need to be updated. The default is 60 seconds.
gns_symavoid	List of Symmetrix IDs (separated by commas) that the daemon should ignore. The daemon does not look for, or place GNS data on, these Symmetrix arrays.
gns_syminclude	List of Symmetrix IDs (separated by commas) that the daemon should manage. If you define this option, GNS uses only these Symmetrix arrays and no others.
gns_remote_mirror	When set to ENABLE, the daemon attempts to mirror RDF group definitions on remote Symmetrix arrays. The default is DISABLE.
gns_shadow_file	When set to ENABLE, the daemon maintains backup copies of GNS data for all local Symmetrix arrays. The default is DISABLE.
logfile_size	Maximum size (in KB) that the GNS daemon's log file is permitted to grow to before wrapping. The default is 1000 KB.
gns_init_remote_symms	If enabled (the default), a local GNS daemon initializes the GNS group repository on remote Symmetrix arrays during remote mirroring (when it detects a repository that needs it). If this init option is disabled, a remote GNS daemon directly connected to such remote arrays has to perform the initialization.

Using a Command to Revise Configuration Settings in the GNS Daemon Options File

Beginning with Solutions Enabler version 6.0, you can use a `stord daemon setoption` command to change or remove a configuration setting in the GNS daemon options file. This feature allows you to make changes to the file programmatically.

For example, you could change the `gns_remote_mirror` option from DISABLE to ENABLE.

```
stord daemon setoption storgnsd -name gns_remote_mirror=enable
```

If you want to remove an option from the file, set the option to a blank value. For example, to remove the `symavoid` option:

```
stord daemon setoption storgnsd -name symavoid=
```

By default, only options already present in the file can be set. In the case where an option does not exist in the file, you can use the `-force` parameter to add a new option line to the file. For example, to add an option called `newoption`:

```
stord daemon setoption storgnsd -name newoption=enable -force
```

The `-force` parameter is used primarily to add an undocumented option that was not provided in the EMC-supplied sample `daemon_options` file. Note that, if you fail to copy `README.daemon_options` to `daemon_options`, you will be starting with an empty GNS options file.

Implementing User Authorization for the GNS Daemon

Because GNS group definitions are shared across users and hosts, a limited form of user authorization is available. You can control access by using the common authorization file (`daemon_users`):

- UNIX: `/var/symapi/config/daemon_users`
- Windows: `c:\Program Files\EMC\SYMAPI\config\daemon_users`

To use and control the GNS daemon, a user must meet one of the following criteria:

- The user is listed in the `daemon_users` file.
- The user is privileged (Unix: root; Windows: member of Administrators group)

The following example entries in the `daemon_users` file allows users named Smith and Jones to access the GNS daemon for use and control:

```
smith      storgnsd
jones      storgnsd
```

Mirroring an RDF Group Definition on a Remote Symmetrix Array

An RDF device group or composite group can contain local devices (either R1 or R2) that are paired with a remote counterpart (R2 or R1). The GNS group definition for an RDF group is stored on all local (directly attached) Symmetrix arrays for which the group contains devices.

You can have GNS automatically maintain a mirrored group definition on the remote Symmetrix array. This mirroring of groups converts a local RDF1 type group to a remote RDF2 type group, and a local RDF2 type group to a remote RDF1 type group. You can use this remotely-mirrored group in a disaster recovery situation where entire applications fail over and are re-started at the remote site.

When you make changes to the local group definition, GNS automatically propagates corresponding changes to the remote group definition.³ Doing this keeps the two group definitions synchronized.

The remotely mirrored group has the same name as the local group and has a mirror image of its contents. Table 2 shows some fields of the local group definition are swapped to ensure that the mirrored remote group is a legal, usable group to hosts.

To enable mirroring of an RDF group definition, set the following option in the GNS daemon options file:

```
storgnsd:gns_remote_mirror=enable
```

An RDF group that has been created by this mirroring mechanism is flagged internally by GNS as being a mirror. By default, these mirrored groups are read-only and cannot be modified or renamed. However, as described above, GNS continues to update them as you change local groups upon which they are based.

You can change the default behavior by setting the SYMAPI_GNS_MIRRORED_GROUP_CONTROL option in the SYMAPI options file to ENABLE (the default value is DISABLE). With this option enabled, if a mirrored group is directly modified (from a host connected to the remote Symmetrix array on which it is defined), the connection is broken between the mirrored group and the local group upon which it is based. It is no longer a mirror, and changes to its base group are no longer propagated to it. If a mirrored group is renamed or deleted, GNS subsequently recreates the mirrored group from its base group.

During operation, a GNS daemon checks the GNS repository on a Symmetrix array and, if it determines that the repository requires initialization, it does so. If remote mirroring is in effect, a local GNS daemon can (by default) initialize a GNS repository on a remote Symmetrix array. However, if a remote GNS daemon is also actively running, groups are being modified on both the local and remote Symmetrix arrays, and shadow files are enabled for local backup, an adverse interaction could occur between the backup/restore mechanism and the remote mirroring mechanism. In this case, you should disable the `gns_init_remote_syms` option in the GNS daemon options file.

With remote initialization disabled, local GNS daemons can initialize only local Symmetrix arrays, and remote GNS daemons can initialize only the remote Symmetrix arrays to which they are directly connected.

³ If a composite group contains either local or remote BCV devices, the BCVs are not mirrored and will not appear in the group created on the remote host.

Table 2. The Local Group Definition Compared to the Remote Group Definition

Local Device Group on Symmetrix 1		Remote Device Group on Symmetrix 2	
Type	RDF1	Type	RDF2
Role	Have a remote mirror	Role	Am a remote mirror
Remote Symm	Sym2	Remote Symm	Sym1
Local RDF Grp	2	Local RDF Grp	5
Remote RDF Grp	5	Remote RDF Grp	2
Devices	0001, 0002, 0003	Devices	0091, 0092, 0093
Remote-Devs	0091, 0092, 0093	Remote-Devs	0001, 0002, 0003
Logical Devs	DEV001, DEV002, DEV003	Logical Devs	DEV001, DEV002, DEV003
BCVs	0010, 0011	BCVs	0020, 0021, 0022
Remote-BCVs	0020, 0021, 0022	Remote-BCVs	0010, 0011
BCV Logical Devs	BCV001, BCV002	BCV Logical Devs	BCV001, BCV002, BCV003
RBCV Logical Devs	RBCV001, RBCV002, RBCV003	RBCV Logical Devs	RBCV001, RBCV002
Vendor	EMC	Vendor	EMC
ModTime	12345	ModTime	12345
GateKeepers	/dev/rdisk/c2t0d2		

Maintaining a Consistency Support Database

With PowerPath or SRDF/A MSC support, you can define an RDF composite group and enable it for RDF consistency protection. If you store such a group within GNS, any change you make to the group (such as adding a device to it) is automatically added to the PowerPath or SRDF/A MSC database on each host that is running PowerPath and GNS, or the RDF daemon and GNS, respectively.

Used with consistency-enabled composite groups, PowerPath or SRDF/A MSC guarantees consistency if there are RDF failures. The GNS daemon automatically updates the local PowerPath or SRDF/A MSC device database when changes are made to the consistency group. Thus, as long as GNS is enabled and the GNS daemons are running on all relevant hosts, these support databases will remain synchronized with device definitions.

When the GNS daemon is managing an RDF consistency group definition on a host, it is important that every SYMAPI application on that host agree that GNS is active. Outside of RDF consistency groups, this is not an issue. However, a problem could occur, for example, if you changed the SYMAPI_USE_GNS setting in the SYMAPI options file after the GNS daemon started running.

Restrictions

While using GNS daemons on multiple hosts, you should be aware of the following restrictions:

- If a SYMAPI application is running in Read/Write (RW), No Modify, Symmetrix Defer mode, group changes are not sent to global GNS until a Symmetrix Commit is performed.
- Do not mix GNS and non-GNS applications on a host (for example, by changing the SYMAPI options file's SYMAPI_USE_GNS setting after the GNS daemon or other applications are already running).
- Beginning with Solutions Enabler version 6.0 and Enginuity version 5x71, a composite group can contain logical device names (aliases) for devices. However, unlike device groups, the logical device names are optional in a consistency group (some devices in the group may have a logical device name, other devices may not). Also, hosts running prior versions of Solutions Enabler have no knowledge of these device alias names and will not provide them when a consistency group is modified.

Resolving Duplicate Group Names

Group names must be unique according to group type (device group or composite group). The GNS daemon enforces this. If users at two different hosts simultaneously create groups of the same type with the same name using devices on different Symmetrix arrays, both attempts might succeed. Because each host's GNS daemon is polling for changes made elsewhere, the daemons might not detect the name conflict until after the fact.

It is also possible for two different groups to be created with the same name on different Symmetrix arrays. A subsequent reconfiguration could then allow the GNS daemon to see both groups.

If a GNS daemon detects duplicate group names, it returns a state flag informing clients of this fact and modifies the group names to be unique by attaching a numeric suffix. For example, duplicate device groups with the name MyDG could be modified in the following way:

```
MyDG#12345678
MyDG#87654321
```

These modified names are returned to the client. While the groups are in this duplicate name state, you can perform only rename or delete operations on the group. For example:

```
symdbg delete MyDG#12345678 -force
symdbg rename MyDG#87654321 MyDG
```

Backing Up and Restoring GNS Data

Currently, GNS group definitions are stored within the Symmetrix cache of individual Symmetrix arrays. These definitions can be lost if either of the following events occurs:

- A Symmetrix array is totally reinitialized.
- Power is lost on a Symmetrix array running Enginuity versions 5x67 or 5x68. With these versions, the area in global cache being used by GNS is not backed up by the Symmetrix File System (SFS).

You can configure a GNS daemon to automatically backup and restore GNS data from Symmetrix arrays to which the daemon is locally attached. Because the maintenance of these backup images can be fairly expensive and impact performance, it is recommended that you *not* enable automatic backup on all GNS daemons. It should suffice to have a Symmetrix array backed up by two or three GNS daemons.

To configure a daemon to *automatically* maintain backup copies of GNS data for locally attached Symmetrix arrays, set the option to ENABLE:

```
storgnsd:gns_shadow_file=enable
```

Backup copies are maintained on the local disk of the host running the GNS daemon and are automatically restored to the GNS repository on a Symmetrix array if the daemon detects that the repository has been erased (for example, a power failure on a Symmetrix array running Enginuity versions 5x67 or 5x68).

Beginning with Solutions Enabler version 6.0, as an alternative to configuring automatic backup, you can use a command to *manually* back up locally attached Symmetrix arrays:

```
storgnsd backup
```

You can also use commands to manually back up and restore *one* Symmetrix array (for example, SID 000187700079). The restore action restores GNS data to the Symmetrix array from the backup file previously created. (Note that the full SID must be specified with `storgnsd` commands.)

```
storgnsd backup -sid 000187700079
storgnsd restore -sid 000187700079
```

Manual backups are probably most useful if automatic backup (`gns_shadow_file`) is disabled, as it is by default. Even if automatic backup is enabled, it may still make sense for you to make periodic copies of a shadow file that is automatically or manually generated, to ensure that a point-in-time copy is available if you need it to manually restore a GNS repository in the future. Because automatic backup regularly overwrites the shadow backup file, the backup version you need may not be available unless you copy it.

The backup shadow file is written to the local disk as follows, where *SID* corresponds to a Symmetrix ID (for example, 000187900035.shadow).

- UNIX: `/var/symapi/gns/SID.shadow`
- Windows: `c:\Program Files\EMC\SYMAPI\gns\SID.shadow`

With automatic backup, the shadow file is overwritten each time a GNS daemon notices a change to group data on the Symmetrix array in question.

Automatic backup and manual backup write to the same shadow file. If you generate a manual backup while the automatic backup mechanism is enabled, the automatic backup will eventually overwrite the manual backup that you generated. Consequently, it is recommended that you make a copy of any backup shadow file that you have generated manually.

Diagnostics

Diagnostic messages are written to the log files `storgnsd.log0` and `storgnsd.log1`, which are located in the `log` directory:

- UNIX: `/var/symapi/log`
- Windows: `c:\Program Files\EMC\SYMAPI\log`

You can display GNS space utilization on each Symmetrix array using the following command:

```
stordaeomon action storgnsd -cmd freespace
```

Although extremely rare, it is possible for the GNS data on a Symmetrix array to appear corrupt if I/O errors occur in the middle of certain critical updates to it. When the log files report that GNS is in a corrupt state, it means that the system encountered an I/O error at a critical point of processing and that the GNS area *might* be in a corrupt state (for example, the free block bitmap may not have been updated when it should have been). If this state occurs, GNS operations to that Symmetrix array will fail.

The presence of any “corrupt” errors are displayed in the log file and also in the output of the `freespace` command. In such cases, you can initiate a “fsck” type repair operation using the `storgnsd repair` command (for example on the Symmetrix array whose SID is 000187700079).

```
storgnsd repair -sid 000187700079
```

The repair operation scans the data already in GNS and fixes a number of inconsistencies that it finds. It reconstructs the free block bitmap, indicating which blocks within the GNS area on that Symmetrix are in use and which are free. Reconstructing certain GNS data like this ensures that it is in a consistent state. You can perform this repair operation while the GNS daemon is already running normally.⁴

Alternatively, if you have a valid GNS shadow backup file, you can perform a restore operation as described previously. A restore operation overwrites GNS with new data from the backup file.

⁴ In extreme circumstances, the `storgnsd reset -sid SymmID` command can be used to fully re-initialize the GNS data area on a given Symmetrix. The full Symmetrix ID must be specified. This command should be used with the utmost care.

Example 1: Importing Device Groups into GNS

The following example was performed using Solutions Enabler version 5.4. Two device groups defined in the SYMAPI database of one host are imported into GNS, which then propagates these device group definitions to another locally attached host running GNS.

The following command uses the vi text editor to open the SYMAPI options file and set the option SYMAPI_USE_GNS to ENABLE. This action enables GNS for all applications on the host.

```
# vi /var/symapi/config/options
```

```
SYMAPI_USE_GNS=ENABLE
```

The following command uses the vi text editor to open the daemon options file and set the option AUTORESTART to ENABLE. Doing this makes use of the watchdog daemon, which automatically restarts the GNS daemon if it crashes. (Beginning with Solutions Enabler version 6.0 and Symmetrix arrays running Enginuity version 5x71, this option is enabled by default.)

```
# vi /var/symapi/config/daemon_options
```

```
storgnsd:autorestart=enabled
```

The `stordaeomon start` command starts the GNS daemon manually. To start this daemon automatically, refer to the section “Starting the GNS Daemon.”

```
# stordaeomon start storgnsd
```

Waiting for daemon to start. This may take several seconds.

The `stordaeomon list` command verifies that the GNS and watchdog daemons are running on the host.

```
# stordaeomon list
```

```
Available Daemons  ('[*]': Currently Running):
```

storapid	EMC Solutions Enabler Base Daemon
[*] storgnsd	EMC Solutions Enabler GNS Daemon
[*] storwatchd	EMC Solutions Enabler Watchdog Daemon
stororad	EMC Solutions Enabler SRM Oracle Daemon
storudbd	EMC Solutions Enabler SRM UDB Daemon
storifmxd	EMC Solutions Enabler SRM Informix Daemon
storsybs11d	EMC Solutions Enabler SRM Sybase 11 Daemon
storsybs12d	EMC Solutions Enabler SRM Sybase 12 Daemon
storsybs12.5d	EMC Solutions Enabler SRM Sybase 12.5 Daemon
storsybs12.5_64d	EMC Solutions Enabler SRM Sybase 12.5 Daemon, 64-bit

The `symdg list` command with the `-inactive` option displays device groups that exist currently in the host's SYMAPI database. From the GNS-enabled viewpoint, these are inactive device groups.

```
# symdg list -inactive
```

I N A C T I V E D E V I C E G R O U P S							
Name	Type	Valid	Symmetrix ID	Devs	Number of		
					GKs	BCVs	VDEVs
79api179regular	REGULAR	N/A	000187700079	23	0	3	0
35api179regular	REGULAR	N/A	000187900035	22	0	44	0

The `symdg activateall` command imports all device groups from the SYMAPI database into GNS. From the GNS-enabled viewpoint, these are now active device groups

```
# symdg activateall -noprompt
```

Activating Device Group: '79api179regular' ... Succeeded.

Activating Device Group: '35api179regular' ... Succeeded.

The `symdg list` command displays device groups from the GNS-enabled viewpoint.

```
# symdg list
```

D E V I C E G R O U P S							
Name	Type	Valid	Symmetrix ID	Devs	Number of		
					GKs	BCVs	VDEVs
79api179regular	REGULAR	Yes	000187700079	23	0	3	0
35api179regular	REGULAR	Yes	000187900035	22	0	44	0

The `symdg show` command displays details of the device group named 79api179regular.

```
# symdg show 79api179regular
```

Group Name: 79api179regular

```

Group Type                : REGULAR
Device Group in GNS       : Yes
Valid                     : Yes
Symmetrix ID              : 000187700079
Group Creation Time       : Thu Nov 20 16:14:16 2003
Vendor ID                 : EMC Corp
Application ID            : SYMCLI

```

```

Number of STD Devices in Group : 23
Number of Associated GK's      : 0
Number of Locally-associated BCV's : 3
Number of Locally-associated VDEV's : 0
Number of Remotely-associated BCV's (STD RDF) : 0
Number of Remotely-associated BCV's (BCV RDF) : 0
Number of Remotely-assoc'd RBCV's (RBCV RDF) : 0

```

Standard (STD) Devices (23):

```

{
-----
LdevName                PdevName                Sym      Att.  Sts      Cap
Dev                    (MB)
-----
DEV001                  /dev/vx/rdmp/c5t0d6s2  0042      RW      2878
DEV002                  /dev/vx/rdmp/c5t0d7s2  0043      RW      2878
DEV003                  /dev/vx/rdmp/c5t0d8s2  0044      RW      2878
DEV004                  /dev/vx/rdmp/c5t0d9s2  0045      RW      2878
DEV005                  /dev/vx/rdmp/c5t0d10s2 0046      RW      2878
DEV006                  /dev/vx/rdmp/c5t0d11s2 0047      RW      2878
DEV007                  /dev/vx/rdmp/c5t0d12s2 0048      RW      2878
DEV008                  /dev/vx/rdmp/c5t0d13s2 0049      RW      2878
DEV009                  /dev/vx/rdmp/c5t0d14s2 004A      RW      2878
DEV010                  /dev/vx/rdmp/c5t0d15s2 004B      RW      2878
DEV011                  /dev/vx/rdmp/c5t0d16s2 004C      RW      2878
DEV012                  /dev/vx/rdmp/c5t0d17s2 004D      RW      2878
DEV013                  /dev/vx/rdmp/c5t0d18s2 004E      RW      2878
DEV014                  /dev/vx/rdmp/c5t0d19s2 004F      RW      2878
DEV015                  /dev/vx/rdmp/c5t0d20s2 0050      RW      2878
DEV016                  /dev/vx/rdmp/c5t0d21s2 0051      RW      2878
DEV017                  /dev/vx/rdmp/c5t0d22s2 0052      RW      2878
DEV018                  /dev/vx/rdmp/c5t0d23s2 0053      RW      2878
DEV019                  /dev/vx/rdmp/c5t0d24s2 0054      RW      2878
DEV020                  /dev/vx/rdmp/c5t0d25s2 0348 (M)  RW      14391
DEV021                  /dev/vx/rdmp/c5t0d26s2 034D (M)  RW      14391
DEV022                  /dev/vx/rdmp/c5t0d27s2 0370 (M)  RW      14391
DEV023                  /dev/vx/rdmp/c5t0d28s2 0375 (M)  RW      14391
}

```

BCV Devices Locally-associated (3):

```

{
-----
LdevName                PdevName                Sym      Att.  Sts      Cap
Dev                    (MB)
-----
BCV001                  N/A                    0454      NR      2878
BCV002                  N/A                    0455      RW      2878
BCV003                  N/A                    0456      RW      2878
}

```

Device Group BCV RDF Information

```

{
RDF Type                : R1
RDF (RA) Group Number   : 3                (02)

Remote Symmetrix ID     : 000187700067

R2 Device Is Larger Than The R1 Device : False

RDF Mode                : Synchronous
RDF Adaptive Copy       : Disabled
RDF Adaptive Copy Write Pending State : N/A
RDF Adaptive Copy Skew (Tracks)       : 65535

RDF Device Domino       : Disabled
}

```

```

RDF Link Configuration           : Fibre
RDF Link Domino                  : Disabled
Prevent Automatic RDF Link Recovery : Disabled
Prevent RAs Online Upon Power ON  : Enabled

Device RDF Status                : Ready          (RW)

Device RA Status                 : Ready          (RW)
Device Link Status               : Not Ready     (NR)

Device Suspend State             : N/A
Device Consistency State         : Disabled
RDF R2 Not Ready If Invalid      : Enabled

Device RDF State                 : Mixed          (MxD)
Remote Device RDF State          : Not Ready     (NR)

RDF Pair State ( R1 <- -> R2 )  : Suspended

Number of R1 Invalid Tracks      : 0
Number of R2 Invalid Tracks      : 0
}

```

The example now enables GNS on another host connected to the same Symmetrix array by setting the SYMAPI_USE_GNS option in this host's SYMAPI options file as was shown previously. After starting the GNS daemon on this host, you can use the `symdg list` command to verify that GNS has propagated the device group definitions to this host.

symdg list

D E V I C E			G R O U P S					
Name	Type	Valid	Symmetrix ID	Devs	Number of			
					GKs	BCVs	VDEVs	
79api179regular	REGULAR	Yes	000187700079	23	0	3	0	
35api179regular	REGULAR	Yes	000187900035	22	0	44	0	

Example 2: Using GNS to Propagate a Composite Group Definition

The following example was performed using Solutions Enabler version 5.4. The hardware setup has two hosts (Host A and Host B) connected to three Symmetrix arrays (SIDs 43, 60, and 35). GNS is enabled and running on both hosts. PowerPath is installed on both hosts.

From Host A, the example creates a composite group definition within GNS, which automatically adds the definition to the PowerPath database on each host. The example then updates the composite group definition from Host B by adding additional devices. GNS automatically propagates the updated composite group definition to the PowerPath database on Host A.

From Host A, the `symcg create` command creates an RDF1 composite group with the name 3-sym-gns. Because GNS is enabled on the host, the group is stored in GNS, which automatically adds it to the PowerPath database on Host A and Host B.

```
# symcg create 3-sym-gns -type rdf1 -ppath
```

The `symcg addall` commands add to the composite group a range of devices from two Symmetrix arrays (SIDs 43 and 60).

```
# symcg -cg 3-sym-gns addall dev -range 110:128 -sid 43
# symcg -cg 3-sym-gns addall dev -range 784:788 -sid 60
```

The `symcg enable` command enables the composite group for RDF consistency protection, which transforms the group into a *consistency group*.

```
# symcg -cg 3-sym-gns enable -noprompt
```

A consistency 'Enable' operation execution is in progress for composite group '3-sym-gns'. Please wait...

The consistency 'Enable' operation successfully executed for composite group '3-sym-gns'.

From Host A, the `symcg list` command with the `-ppath` option displays those composite groups that currently exist in the PowerPath database on Host A. Only the newly-created 3-sym-gns group currently exists. GNS has propagated this composite definition to Host B (as shown later).

```
# symcg list -ppath
```

C O M P O S I T E			G R O U P S				
Name	Type	Valid	Number of		Number of		
			Symms	RAGs	Devs	BCVs	VDEVs
3-sym-gns	RDF1	Yes	2	2	27	0	0

The `symcg show` command displays details of the composite group. Notice that the composite group (CG) is in PowerPath and in GNS as indicated by the "Yes" in those rows.

```
# symcg show 3-sym-gns
```

```
Composite Group Name: 3-sym-gns
```

```
Composite Group Type      : RDF1
Valid                    : Yes
CG in PowerPath          : Yes
CG in GNS                : Yes
```

```
Number of RDF (RA) Groups      : 2
Number of STD Devices         : 27
Number of BCV's (Locally-associated) : 0
Number of VDEV's (Locally-associated) : 0
Number of RBCV's (Remotely-associated STD-RDF) : 0
Number of BRBCV's (Remotely-associated BCV-RDF) : 0
Number of RRBCV's (Remotely-associated RBCV) : 0
```

```
Number of Symmetrix Units (2):
```

```
{
  1) Symmetrix ID              : 000000003143
     Microcode Version         : 5267
     Number of STD Devices     : 22
     Number of BCV's (Locally-associated) : 0
     Number of VDEV's (Locally-associated) : 0
     Number of RBCV's (Remotely-associated STD_RDF) : 0
     Number of BRBCV's (Remotely-associated BCV-RDF) : 0
     Number of RRBCV's (Remotely-associated RBCV) : 0
```

```
Number of RDF (RA) Groups (1):
```

```
{
  1) RDF (RA) Group Number : 1          (A)
     Remote Symmetrix ID   : 000000003156
     Microcode Version     : 5267
```

```
STD Devices (22):
```

```
{
-----
PdevName          Sym  Device  Consistency  Cap
                  Dev  Config  State        (MB)
-----
/dev/vx/rdmp/c3t3d0s2  0110 RDF1    Enabled     12946
/dev/vx/rdmp/c3t3d1s2  0113 RDF1    Enabled     8631
/dev/vx/rdmp/c3t3d2s2  0115 RDF1    Enabled     4315
/dev/vx/rdmp/c3t3d3s2  0116 RDF1    Enabled     4315
/dev/vx/rdmp/c3t3d4s2  0117 RDF1    Enabled     4315
/dev/vx/rdmp/c3t3d5s2  0118 RDF1    Enabled     4315
/dev/vx/rdmp/c3t3d6s2  0119 RDF1    Enabled     4315
/dev/vx/rdmp/c3t3d7s2  011A RDF1    Enabled     4315
/dev/vx/rdmp/c3t4d0s2  011B RDF1    Enabled     4315
/dev/vx/rdmp/c3t4d1s2  011C RDF1    Enabled     4315
/dev/vx/rdmp/c3t4d2s2  011D RDF1    Enabled     4315
-----
```

```

        /dev/vx/rdmp/c3t4d3s2    011E RDF1      Enabled    4315
        /dev/vx/rdmp/c3t4d4s2    011F RDF1      Enabled    4315
        /dev/vx/rdmp/c3t4d5s2    0120 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t4d6s2    0121 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t4d7s2    0122 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d0s2    0123 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d1s2    0124 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d2s2    0125 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d3s2    0126 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d4s2    0127 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d5s2    0128 RDF1      Enabled    4315
    }
}

2) Symmetrix ID                      : 000184500160
   Microcode Version                  : 5568
   Number of STD Devices               : 5
   Number of BCV's (Locally-associated) : 0
   Number of VDEV's (Locally-associated) : 0
   Number of RBCV's (Remotely-associated STD_RDF) : 0
   Number of BRBCV's (Remotely-associated BCV-RDF) : 0
   Number of RRBCV's (Remotely-associated RBCV) : 0

Number of RDF (RA) Groups (1):
{
    1) RDF (RA) Group Number : 1 (A)
       Remote Symmetrix ID : 000184502898
       Microcode Version : 5568

       STD Devices (5):
       {
           -----
           PdevName          Sym Device Consistency Cap
                           Dev Config State (MB)
           -----
           /dev/vx/rdmp/c5t2d24s2  0784 RDF1      Enabled    4315
           /dev/vx/rdmp/c5t2d25s2  0785 RDF1      Enabled    4315
           /dev/vx/rdmp/c5t2d26s2  0786 RDF1      Enabled    4315
           /dev/vx/rdmp/c5t2d27s2  0787 RDF1      Enabled    4315
           /dev/vx/rdmp/c5t2d28s2  0788 RDF1      Enabled    4315
       }
    }
}

```

Now from Host B, issuing the `symcg list` command with the `-ppath` option displays the composite group that currently exists in the PowerPath database on Host B. The display reveals the same composite group that exists on Host A.

```
# symcg list -ppath
```

C O M P O S I T E G R O U P S								
Name	Type	Valid	Number of		Number of			
			Symms	RAGs	Devs	BCVs	VDEVs	
3-sym-gns	RDF1	Yes	2	2	27	0	0	

Also from Host B, the `symcg addall` command now adds to this composite group a range of devices from the third Symmetrix array (SID 35).

```
# symcg -cg 3-sym-gns addall dev -range 150:168 -sid 35
```

Returning to Host A and issuing the `symcg list` command there displays whether the changes made to the composite group definition from Host B have been propagated to Host A. (You may have to wait up to 60 seconds to see the changes propagated.) The following display shows the addition of the third Symmetrix array and its devices to the composite group.

```
# symcg list -ppath
```

C O M P O S I T E G R O U P S								
Name	Type	Valid	Number of		Number of			
			Symms	RAGs	Devs	BCVs	VDEVs	
3-sym-gns	RDF1	Yes	3	3	49	0	0	

The `symcg show` command displays the updated details of the composite group. Note that the newly added devices from Symmetrix 000187900035 are not yet enabled for RDF consistency protection.

```
# symcg show 3-sym-gns
```

```
Composite Group Name: 3-sym-gns
```

```
Composite Group Type      : RDF1
Valid                     : Yes
CG in PowerPath           : Yes
CG in GNS                 : Yes
```

```
Number of RDF (RA) Groups : 3
Number of STD Devices     : 49
Number of BCV's (Locally-associated) : 0
Number of VDEV's (Locally-associated) : 0
Number of RBCV's (Remotely-associated STD-RDF) : 0
Number of BRBCV's (Remotely-associated BCV-RDF) : 0
Number of RRBCV's (Remotely-associated RBCV) : 0
```

```
Number of Symmetrix Units (3):
```

```
{
    1) Symmetrix ID : 000187900035
```

```

Microcode Version           : 5670
Number of STD Devices       : 22
Number of BCV's (Locally-associated) : 0
Number of VDEV's (Locally-associated) : 0
Number of RBCV's (Remotely-associated STD_RDF) : 0
Number of BRBCV's (Remotely-associated BCV-RDF) : 0
Number of RRBCV's (Remotely-associated RBCV) : 0

```

Number of RDF (RA) Groups (1):

{

```

1) RDF (RA) Group Number : 1          (00)
   Remote Symmetrix ID   : 000187900041
   Microcode Version     : 5670

```

STD Devices (22):

{

PdevName	Sym Dev	Device Config	Consistency State	Cap (MB)
/dev/vx/rdmp/c6t0d24s2	0150	RDF1+Mir	Disabled	12946
/dev/vx/rdmp/c6t0d25s2	0153	RDF1+Mir	Disabled	8631
/dev/vx/rdmp/c6t0d26s2	0155	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d27s2	0156	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d28s2	0157	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d29s2	0158	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d30s2	0159	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d31s2	015A	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d32s2	015B	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d33s2	015C	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d34s2	015D	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d35s2	015E	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d36s2	015F	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d37s2	0160	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d38s2	0161	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d39s2	0162	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d40s2	0163	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d41s2	0164	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d42s2	0165	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d43s2	0166	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d44s2	0167	RDF1+Mir	Disabled	4315
/dev/vx/rdmp/c6t0d45s2	0168	RDF1+Mir	Disabled	4315

}

}

```

2) Symmetrix ID           : 000184500160
   Microcode Version       : 5568
   Number of STD Devices   : 5
   Number of BCV's (Locally-associated) : 0
   Number of VDEV's (Locally-associated) : 0
   Number of RBCV's (Remotely-associated STD_RDF) : 0
   Number of BRBCV's (Remotely-associated BCV-RDF) : 0
   Number of RRBCV's (Remotely-associated RBCV) : 0

```

Number of RDF (RA) Groups (1):

{

```

1) RDF (RA) Group Number : 1 (A)
   Remote Symmetrix ID    : 000184502898
   Microcode Version      : 5568

```

```
STD Devices (5):
```

```

{
-----
PdevName          Sym  Device  Consistency  Cap
                  Dev  Config  State        (MB)
-----
/dev/vx/rdmp/c5t2d24s2 0784 RDF1    Enabled      4315
/dev/vx/rdmp/c5t2d25s2 0785 RDF1    Enabled      4315
/dev/vx/rdmp/c5t2d26s2 0786 RDF1    Enabled      4315
/dev/vx/rdmp/c5t2d27s2 0787 RDF1    Enabled      4315
/dev/vx/rdmp/c5t2d28s2 0788 RDF1    Enabled      4315
}
}

```

```

3) Symmetrix ID          : 000000003143
   Microcode Version      : 5267
   Number of STD Devices  : 22
   Number of BCV's (Locally-associated) : 0
   Number of VDEV's (Locally-associated) : 0
   Number of RBCV's (Remotely-associated STD_RDF) : 0
   Number of BRBCV's (Remotely-associated BCV-RDF) : 0
   Number of RRBCV's (Remotely-associated RBCV) : 0

```

```
Number of RDF (RA) Groups (1):
```

```

{
1) RDF (RA) Group Number : 1 (A)
   Remote Symmetrix ID    : 000000003156
   Microcode Version      : 5267

```

```
STD Devices (22):
```

```

{
-----
PdevName          Sym  Device  Consistency  Cap
                  Dev  Config  State        (MB)
-----
/dev/vx/rdmp/c3t3d0s2 0110 RDF1    Enabled      12946
/dev/vx/rdmp/c3t3d1s2 0113 RDF1    Enabled       8631
/dev/vx/rdmp/c3t3d2s2 0115 RDF1    Enabled      4315
/dev/vx/rdmp/c3t3d3s2 0116 RDF1    Enabled      4315
/dev/vx/rdmp/c3t3d4s2 0117 RDF1    Enabled      4315
/dev/vx/rdmp/c3t3d5s2 0118 RDF1    Enabled      4315
/dev/vx/rdmp/c3t3d6s2 0119 RDF1    Enabled      4315
/dev/vx/rdmp/c3t3d7s2 011A RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d0s2 011B RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d1s2 011C RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d2s2 011D RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d3s2 011E RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d4s2 011F RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d5s2 0120 RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d6s2 0121 RDF1    Enabled      4315
/dev/vx/rdmp/c3t4d7s2 0122 RDF1    Enabled      4315
}

```

```

        /dev/vx/rdmp/c3t5d0s2    0123 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d1s2    0124 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d2s2    0125 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d3s2    0126 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d4s2    0127 RDF1      Enabled    4315
        /dev/vx/rdmp/c3t5d5s2    0128 RDF1      Enabled    4315
    }
}

```

Another `symcg enable` command enables the remaining devices in the composite group for RDF consistency protection.

```
# symcg -cg 3-sym-gns enable -noprompt
```

A consistency 'Enable' operation execution is in progress for composite group '3-sym-gns'. Please wait...

The consistency 'Enable' operation successfully executed for composite group '3-sym-gns'.