IBM Storage Scale RAID
5.1.8

Administration

SC28-3452-01

Note

Before using this information and the product it supports, read the information in "Notices" on page 441.

This edition applies to Version 5 release 1 modification 8 of the following products and to all subsequent releases and modifications until otherwise indicated in new editions:

- IBM Spectrum® Scale Data Management Edition for IBM® ESS (product number 5765-DME)
- IBM Storage Scale Data Access Edition for IBM ESS (product number 5765-DAE)

IBM welcomes your comments; see the topic "How to submit your comments" on page xiv. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this information

This information is intended as a guide for administering IBM Storage Scale RAID on Power Systems servers.

Who should read this information

This information is intended for administrators of systems on Power Systems servers that include IBM Storage Scale RAID.

Conventions used in this information

Table 1 on page xiii describes the typographic conventions used in this information. UNIX file name conventions are used throughout this information.

Table 1. Conventions

Convention	Usage
bold	Bold words or characters represent system elements that you must use literally, such as commands, flags, values, and selected menu options. Depending on the context, **bold** typeface sometimes represents path names, directories, or file names.
bold underlined	bold underlined keywords are defaults. These take effect if you do not specify a different keyword.
constant width	Examples and information that the system displays appear in `constant-width` typeface. Depending on the context, `constant-width` typeface sometimes represents path names, directories, or file names.
italic	*Italic* words or characters represent variable values that you must supply. *Italics* are also used for information unit titles, for the first use of a glossary term, and for general emphasis in text.
<*key*>	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, <Enter> refers to the key on your terminal or workstation that is labeled with the word *Enter*.

****	In command examples, a backslash indicates that the command or coding example continues on the next line. For example:
	`mkcondition -r IBM.FileSystem -e "PercentTotUsed > 90" \`
	`-E "PercentTotUsed < 85" -m p "FileSystem space used"`

{*item*}	Braces enclose a list from which you must choose an item in format and syntax descriptions.
[*item*]	Brackets enclose optional items in format and syntax descriptions.
<Ctrl-*x*>	The notation <Ctrl-*x*> indicates a control character sequence. For example, <Ctrl-c> means that you hold down the control key while pressing <c>.
***item*...**	Ellipses indicate that you can repeat the preceding item one or more times.

Table 1. Conventions (continued)

Convention	Usage
|	In *synopsis* statements, vertical lines separate a list of choices. In other words, a vertical line means *Or*. In the left margin of the document, vertical lines indicate technical changes to the information.

How to submit your comments

Your feedback is important in helping us to produce accurate, high-quality information. You can add comments about this information in IBM Documentation.

To contact the IBM Storage Scale development organization, send your comments to the following email address:

`scale@us.ibm.com`

Summary of changes

This topic summarizes changes to the IBM Storage Scale RAID information. A vertical line (|) to the left of text and illustrations indicates technical changes or additions made to the previous edition of the information.

Summary of changes
for IBM Storage
Scale RAID version 5.1.8
as updated, July 2023

Changes to this release include the following:

Administering

- The ESS fabric hospital support added to monitor SAS-based RAS events on a fabric storage. For more information, see “ESS fabric hospital” on page 11.

Commands

The following lists the modifications to the commands:

- **mmvdisk**
- **mmvdisk nvmeof**

Management API changes

No APIs were added or modified

Management GUI changes

No changes were added or modified

New messages

No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.7
as updated, March 2023

Changes to this release include the following:

Administering

No changes were added or modified

Commands

The following lists the modifications to the commands:

- **mmvdisk nodeclass**
- **mmvdisk server**
- **mmvdisk recoverygroup**
- **mmvdisk filesystem**

Management API changes

No APIs were added or modified

Management GUI changes

- The Hardware page includes the status of non-stretch clusters and which enclosures are daisy chained.
- On the Declustered Arrays page, the compression rate is displayed along with the physical and total capacities.

New messages
No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.6
as updated, March 2023

Changes to this release include the following:

Administering

Commands
The following lists the modifications to the commands:

- **mmvdisk recoverygroup**

Management API changes
No API's were added or modified

Management GUI changes
No GUI changes.

New messages
No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.5
as updated, November 2022

Changes to this release include the following:

Administering

Commands
The following lists the modifications to the commands:

- **mmlspdisk**
- **mmvdisk**
- **mmvdisk sed**

Management API changes
No API's were added or modified

Management GUI changes
The IBM FlashCore® Module (FCM) specified space and health information that is provided by the ZIMon collector is now displayed on the Declustered Array **Dashboard** page in the GUI.

New messages
No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.4
as updated, June 2022

Changes to this release include the following:

Administering
Support provided for automatic background TRIM.

Commands
No commands have been changed

Management API changes
No API's were added or modified

Management GUI changes
All open support tickets can be viewed and managed on the Call Home page in the ESS 3500 GUI .

New messages
No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.3
as updated, March 2022

Changes to this release include the following:

Commands
No commands have been changed

Management API changes
Modified the following API endpoints.

- PUT gnr/recoverygroups/pdisk/change
- PUT gnr/recoverygroups/pdisk/replace

New messages
No new messages have been added

Summary of changes
for IBM Storage
Scale RAID version 5.1.2
as updated, July 2021

Changes to this release include the following:

Commands
The following lists the modifications to the commands:

Changed commands

- **mmchrecoverygroup**
- **mmlspdisk**
- **mmlsrecoverygroup**
- **mmvdisk recoverygroup**

Management API changes
Modified the following API endpoints.

- GET Gnr/diskmgmt/vdiskset/
- POST gnr/diskmgmt/vdiskset/define
- GET encryption/clients
- GET encryption/keys
- GET encryption/servers
- GET encryption/tenants
- POST encryption/servers
- POST encryption/tenants
- POST encryption/keys
- POST encryption/clients
- POST encryption/client/register
- PUT encryption/client/deregister
- PUT encryption/clients

- PUT encryption/servers
- DELETE encryption/tenants
- DELETE encryption/clients/{clientName}
- DELETE encryption/servers/{serverName}

New messages

6027-3867, 6027-3868, 6027-3869, and 6027-3870, 6027-3871, 6027-3872, and 6027-3873

Summary of changes for IBM Storage Scale RAID version 5.1.1.3 as updated, August 2021

Changes to this release include the following:

Management API changes

Added the following API endpoint.

- GET Gnr/diskmgmt/vdiskset/server/vdiskList

Modified the following API endpoints.

- PUT Gnr/recoverygroups/pdiskChange/change
- POST Gnr/diskmgmt/vdiskset/define

Summary of changes for IBM Storage Scale RAID version 5.1.1.2 as updated, July 2021

Changes to this release include the following:

Commands

The following lists the modifications to the commands:

Changed commands

mmvdisk recoverygroup

Management API changes

Added the following API endpoints.

- GET gnr/diskmgmt/vdiskset/server/list/*{nodeClass}*
- GET gnr/recoverygroups/declusteredArray
- GET gnr/recoverygroups/*{rgName}*/*{oldNode}*/replace/*{newNode}*
- GET encryption/rkmClients
- GET encryption/rkmKeys
- GET encryption/rkmServer
- GET encryption/rkms
- GET encryption/rkmTenants
- POST encryption/deregisterClient
- POST encryption/updateClientkeys
- POST encryption/serverAdd
- POST encryption/addTenant
- POST encryption/createKey
- POST encryption/clients
- POST encryption/registerClient

- PUT encryption/updateClientkeys
- PUT encryption/serverUpdate
- PUT recoverygroups/pdiskReplace/replace
- PUT disk/Change
- DELETE encryption/clientName/*{clientName}*
- DELETE encryption/deleteTenant
- DELETE encryption/serverName/*{serverName}*

Modified the following API endpoint.

- GET gnr/clustermgmt/nodes/{names}/state

Summary of changes
for IBM Storage
Scale RAID version 5.1.1.1
as updated, June 2021

Changes to this release include the following:

Commands

The following lists the modifications to the commands:

Changed commands

`mmvdisk recoverygroup`

Management API changes

Added the following API endpoints.

- GET gnr/clustermgmt /state
- PUT pdiskReplace/replace
- PUT gnr/recoverygroups/suspend/{recoveryGroupName}/{node}
- PUT gnr/recoverygroups/resume/{recoveryGroupName}/{node}
- POST filesystems/{filesystem}/file/{path}/fileSize/{size}
- POST clustermgmt/server/configure
- POST gnr/diskmgmt/vdiskset/define
- POST gnr/recoverygroups/recoveryGroupName/node/{name}
- POST gnr/diskmgmt/vdiskset/create/{vdisk Set}
- POST gnr/diskmgmt/vdiskset/vdiskSet/rename/{vdiskSet}
- POST gnr/diskmgmt/vdisk set/{vdiskSet}/rename/{newVdiskSet}
- POST filesystems/filesystemName/vdiskSet/vdiskSet
- POST gnr/clustermgmt/{newNodeName}/service/{serviceName}
- DELETE filesystems/{filesystemName}/file/{path}
- DELETE gnr/diskmgmt/vdiskset/undefine/{vdiskSet}
- DELETE gnr/diskmgmt/vdiskset/delete/{vdiskSet}

Summary of changes
for IBM Storage
Scale RAID version 5.1.1
as updated, April 2021

Changes to this release include the following:

Commands

The following lists the modifications to the commands:

Changed commands
`mmvdisk recoverygroup`

Summary of changes
for IBM Storage
Scale RAID version 5.0.5.5
as updated, December 2020

Changes to this release include the following:

Commands
The following lists the modifications to the commands:

Changed commands
`mmvdisk recoverygroup`

Summary of changes
for IBM Storage
Scale RAID version 5.0.5.2
as updated, August 2020

Changes to this release include the following:

Commands
The following lists the modifications to the commands:

Changed commands
The following commands, command descriptions, or both were changed:

- **`mmvdisk recoverygroup`**

Summary of changes
for IBM Storage
Scale RAID version 5.0.5.1
as updated, July 2020

Changes to this release include the following:

Administering
Support provided for TRIM

Summary of changes
for IBM Storage
Scale RAID version 5.0.4.3
as updated, March 2020

Changes to this release include the following:

New command
`mmlsnvmestatus`

New messages
`6027-3859, 6027-3860, 6027-3861, and 6027-3862`

Summary of changes
for IBM Storage
Scale RAID version 5.0.4.1
as updated, December 2019

Changes to this release include the following:

Commands
The following lists the modifications to the commands:

Changed commands
The following commands, command descriptions, or both were changed:

- **mmchenclosure**
- **mmlsenclosure**

Summary of changes
for IBM Storage
Scale RAID version 5.0.4
as updated, October 2019

Changes to this release include the following:

Managing IBM Storage Scale RAID with the mmvdisk command

mmvdisk now supports custom spare pdisk settings for the declustered arrays of scale-out recovery groups, using the **mmvdisk recoverygroup change --spare-pdisks** or **mmvdisk recoverygroup change --spare-nodes** commands

mmvdisk now supports safely suspending a server in a scale-out recovery group so that server maintenance can be performed, and resuming the server when maintenance is complete, using the **mmvdisk recoverygroup change --suspend** and **mmvdisk recoverygroup change --resume** commands.

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- **mmvdisk recoverygroup**

Summary of changes
for IBM Storage
Scale RAID version 5.0.3
as updated, June 2019

Changes to this release include the following:

Managing IBM Storage Scale RAID with the mmvdisk command

This section has been updated with additional information about node class, server, recovery group, vdisk set, and file system management.

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- **mmvdisk**
- **mmvdisk nodeclass**
- **mmvdisk server**
- **mmvdisk recoverygroup**
- **mmvdisk filesystem**
- **mmvdisk vdiskset**
- **mmvdisk vdisk**
- **mmvdisk pdisk**

Summary of changes
for IBM Storage
Scale RAID version 5.0.2
as updated, November 2018

Changes to this release include the following:

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- **gnrhealthcheck**
- **mmlspdisk**
- **mmvdisk**
- **mmvdisk nodeclass**
- **mmvdisk server**
- **mmvdisk recoverygroup**
- **mmvdisk vdiskset**
- **mmvdisk filesystem**
- **mmvdisk pdisk**
- **mmvdisk vdisk**

Messages

New message:

- 6027-3850

Summary of changes for IBM Storage Scale RAID version 5.0.1.1 as updated, June 2018

Changes to this release include the following:

Managing IBM Storage Scale RAID with the mmvdisk command

A new section has been added that introduces the **mmvdisk** command for managing IBM Storage Scale RAID.

The **mmvdisk** command simplifies IBM Storage Scale RAID administration. It also enforces and encourages consistent best practices for IBM Storage Scale RAID servers, recovery groups, vdisk NSDs, and file systems.

Commands

The following lists the modifications to the commands:

New commands

The following commands are new:

- **mmvdisk**
- **mmvdisk nodeclass**
- **mmvdisk server**
- **mmvdisk recoverygroup**
- **mmvdisk vdiskset**
- **mmvdisk filesystem**
- **mmvdisk pdisk**
- **mmvdisk vdisk**

Changed commands

The following commands, command descriptions, or both were changed:

- **mmchpdisk**

GUI changes

The following are the main changes in this release:

- Prior to the ESS 5.3.1 release, the Create File System wizard in the **Files** > **File Systems** page supported creating file systems with a system pool that can store only metadata. You can now configure the system pool for storing both data and metadata while creating file systems.
- On mmvdisk-enabled ESS clusters, the GUI disk replacement procedure uses the **mmvdisk** command instead of the **mmchcarrier** command.

Messages
No changes.

Summary of changes for IBM Storage Scale RAID version 5.0.0 as updated, March 2018

Changes to this release include the following:

IBM Storage Scale RAID management API
Added the following API endpoints:

- GET /gnr/recoverygroups
- GET /gnr/recoverygroups/{recoveryGroupName}
- GET /gnr/recoverygroups/{recoveryGroupName}/pdisks
- GET /gnr/recoverygroups/{recoveryGroupName}/pdisks/{pdiskName}
- GET /gnr/recoverygroups/{recoveryGroupName}/vdisks
- GET /gnr/recoverygroups/{recoveryGroupName}/vdisks/{vdiskName}

For more information on the newly added endpoints, see Chapter 11, “IBM Storage Scale RAID management API,” on page 129.

Commands
The following lists the modifications to the commands:

Changed commands
The following commands, command descriptions, or both were changed:

- **mmchfirmware**
- **mmchpdisk**

Messages
The following new messages are added in this release:

New messages

6027-3845, 6027-3846, 6027-3847, 6027-3848, and 6027-3849

Summary of changes for IBM Storage Scale RAID version 4.2.3 as updated, May 2017

Changes to this release include the following:

Commands
The following lists the modifications to the commands:

Changed commands
The following commands, command descriptions, or both were changed:

- mmchfirmware
- mmchpdisk

Summary of changes for IBM Storage

Scale RAID version 4.2.2
as updated, January 2017

Changes to this release include the following:

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- `mmchfirmware`
- `mmlsfirmware`

Scripts

The following lists the modifications to the scripts:

Changed scripts

The following scripts, script descriptions, or both were changed:

- `mkrginput`
- `topselect`
- `topsummary`

Messages

The following lists the new messages that are added in this release:

New messages

6027-3812, 6027-3813, 6027-3814, 6027-3815, 6027-3816, 6027-3817, 6027-3818, 6027-3819, 6027-3820, 6027-3821, 6027-3822, 6027-3823, 6027-3824, 6027-3825, 6027-3826, 6027-3827, 6027-3828, 6027-3829, 6027-3830, 6027-3831, 6027-3832, 6027-3833, 6027-3836, 6027-3837, 6027-3838, 6027-3839, 6027-3840, 6027-3841, 6027-3843, 6027-3844

Changed messages

6027-1860, 6027-1861, 6027-1864, 6027-1876, 6027-3016, 6027-3041, 6027-3091, 6027-3092, 6027-3802, 6027-3804, 6027-3805, 6027-3806, 6027-3807, 6027-3808, 6027-3809, 6027-3811

Deleted messages

6027-1859, 6027-3803

Documentation changes

The following main documentation updates are done in this release:

- Modified the monitoring information to add detailed documentation on system health monitoring, performance monitoring, and capacity monitoring.
- Restructured the Troubleshooting section to add more information about the troubleshooting procedures and also to align the troubleshooting information with the standards set for other IBM storage products.

Summary of changes
for IBM Storage
Scale RAID version 4 release 5.0
as updated, August 2016

Changes to this release of IBM Storage Scale RAID include the following:

ESS core

- IBM Storage Scale RAID version 4.2.1-0 efix2
- Updated GUI

- Changes from ESS 4.0.x

Support of Red HatEnterprise Linux® 7.1

- No changes from ESS 3.0, 3.5, and 4.0 (see those sections in this document for more information).
- Change in ESS (same as ESS 4.0.5) kernel release 3.10.0-229.34.1.el7.ppc64

Support of MLNX_OFED_LINUX-3.3-1.0.4.1

- Updated from MLNX_OFED_LINUX-3.2-2.0.0.1 (ESS 4.0.3, 4.0.5)
- Updated from MLNX_OFED_LINUX-3.1-1.0.6.1 (ESS 4.0, 4.0.1, 4.0.2)
- Updated from MLNX_OFED_LINUX-3.1-1.0.0.2 (ESS 3.5.x)
- Updated from MLNX_OFED_LINUX-2.4-1.0.2 (ESS 3.0.x)
- Support for PCIe3 LP 2-port 100 Gb EDR InfiniBand adapter x16 (FC EC3E)
  - Requires System FW level FW840.20 (SV840_104)
  - No changes from ESS 4.0.3

Install Toolkit

- Updated Install Toolkit
- Updates from ESS 4.0.x

Updated firmware rpm

- Updated firmware for IBM PCIe x8 Cache SAS RAID Internal Adapter
- Support for updated drive FW
- Changes from ESS 4.0.5

Summary of changes for IBM Storage Scale RAID version 4 release 2.0.3 as updated, May 2016

Changes to this release of IBM Storage Scale RAID include the following:

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- `mmdelcomploc`
- `mmlsfirmware`

Scripts

The following lists the modifications to the scripts:

Changed scripts

The following scripts, script descriptions, or both were changed:

- `gnrhealthcheck`

Summary of changes for IBM Storage Scale RAID version 4 release 2.0 as updated, January 2016

Changes to this release of IBM Storage Scale RAID include the following:

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- `mmchcarrier`
- `mmchrecoverygroup`
- `mmcrrecoverygroup`
- `mmlsfirmware`
- `mmlspdisk`

Messages

The following lists the new messages that are added in this release:

New messages

6027-3801, 6027-3802, 6027-3803, 6027-3804, 6027-3805, 6027-3806, 6027-3807, 6027-3808, 6027-3809, 6027-3810

Summary of changes for IBM Storage Scale RAID version 4 release 1.1 as updated, October 2015

Changes to this release of IBM Storage Scale RAID, the IBM Storage Scale RAID information, or both include the following:

The ability to create recovery group stanza files with a single data (non-log) declustered array

Commands

The following lists the modifications to the commands:

Changed commands

The following commands, command descriptions, or both were changed:

- `mmchenclosure`
- `mmlsenclosure`
- `mmlsfirmware`

Scripts

The following lists the modifications to the scripts:

Changed scripts

The following scripts, script descriptions, or both were changed:

- `mkrginput`

Messages

The following lists the changed messages:

Changed messages

The following messages, message descriptions, or both were changed:

6027-3045, 6027-3046, MS0103, MS0104, MS0105, MS0241, MS0242, MS0243

Summary of changes for GPFS Native RAID version 4 release 1.0.8 as updated, May 2015

Changes to this release of GNR, the GNR information, or both include the following:

Documented commands

The following lists the modifications to the documented commands:

New commands

The following commands are new:

- mmaddcompspec
- mmdelcompspec

Changed commands

The following commands, command descriptions, or both were changed:

- mmaddcomp
- mmchcomp
- mmchcomploc
- mmchfirmware
- mmchrecoverygroup
- mmcrrecoverygroup
- mmdelcomp
- mmdelcomploc
- mmdiscovercomp
- mmlscomp
- mmlscomploc
- mmlscompspec
- mmlsfirmware
- mmlsrecoverygroup
- mmsyncdisplayid

Messages

The following lists the new and changed messages:

New messages

6027-3095, 6027-3096, 6027-3097, 6027-3098, 6027-3099, 6027-3800

Changed messages

The following messages, message descriptions, or both were changed:

6027-3045, 6027-3046

Chapter 1. Introducing IBM Storage Scale RAID

This topic describes the basic concepts, features, and functions of IBM Storage Scale RAID: redundancy codes, end-to-end checksums, data declustering, and administrator configuration, including recovery groups, declustered arrays, virtual disks, and virtual disk NSDs.

IBM Storage Scale RAID is a software implementation of storage RAID technologies within IBM Storage Scale. Using conventional dual-ported disks in a JBOD configuration, IBM Storage Scale RAID implements sophisticated data placement and error-correction algorithms to deliver high levels of storage reliability, availability, and performance. Standard GPFS file systems are created from the NSDs defined through IBM Storage Scale RAID.

IBM Storage Scale RAID is available with the IBM Elastic Storage® Server (ESS) 5.1 for Power®. ESS is a high-capacity, high-performance storage solution that combines IBM Power Systems servers, storage enclosures, drives, software (including IBM Storage Scale RAID), and networking components. ESS uses a building-block approach to create highly scalable storage for use in a broad range of application environments.

Overview

IBM Storage Scale RAID integrates the functionality of an advanced storage controller into the GPFS NSD server. Unlike an external storage controller, where configuration, LUN definition, and maintenance are beyond the control of IBM Storage Scale, IBM Storage Scale RAID itself takes on the role of controlling, managing, and maintaining physical disks - hard disk drives (HDDs) and solid-state drives (SSDs).

Sophisticated data placement and error correction algorithms deliver high levels of storage reliability, availability, serviceability, and performance. IBM Storage Scale RAID provides a variation of the GPFS network shared disk (NSD) called a *virtual disk* or *vdisk*. Standard NSD clients transparently access the vdisk NSDs of a file system by using the conventional NSD protocol.

The features of IBM Storage Scale RAID include:

- **Software RAID**

  IBM Storage Scale RAID, which runs on standard Serial Attached SCSI (SAS) disks in a dual-ported JBOD array, does not require external RAID storage controllers or other custom hardware RAID acceleration.

- **Declustering**

  IBM Storage Scale RAID distributes client data, redundancy information, and spare space uniformly across all disks of a JBOD. This approach reduces the rebuild (disk failure recovery process) overhead and improves application performance compared to conventional RAID.

- **Pdisk-group fault tolerance**

  In addition to declustering data across disks, IBM Storage Scale RAID can place data and parity information to protect against groups of disks that, based on characteristics of a disk enclosure and system, might possibly fail together due to a common fault. The data placement algorithm ensures that even if all members of a disk group fail, the error correction codes are still capable of recovering erased data.

- **Checksum**

  An end-to-end data integrity check, by using checksums and version numbers, is maintained between the disk surface and NSD clients. The checksum algorithm uses version numbers to detect silent data corruption and lost disk writes.

- **Data redundancy**

  IBM Storage Scale RAID supports highly reliable 2-fault-tolerant and 3-fault-tolerant Reed-Solomon-based parity codes and 3-way and 4-way replication.

- **Large cache**

  A large cache improves read and write performance, particularly for small I/O operations.
- **Arbitrarily-sized disk arrays**

  The number of disks is not restricted to a multiple of the RAID redundancy code width, which allows flexibility in the number of disks in the RAID array.
- **Multiple redundancy schemes**

  One disk array can support vdisks with different redundancy schemes, for example Reed-Solomon and replication codes.
- **Disk hospital**

  A disk hospital asynchronously diagnoses faulty disks and paths, and requests replacement of disks by using past health records.
- **Automatic recovery**

  Seamlessly and automatically recovers from primary server failure.
- **Disk scrubbing**

  A disk scrubber automatically detects and repairs latent sector errors in the background.
- **Familiar interface**

  Standard IBM Storage Scale command syntax is used for all configuration commands, including maintaining and replacing failed disks.
- **Flexible hardware configuration**

  Support of JBOD enclosures with multiple disks physically mounted together on removable carriers.
- **Journaling**

  For improved performance and recovery after a node failure, internal configuration and small-write data are journaled to solid-state disks (SSDs) in the JBOD or to non-volatile random-access memory (NVRAM) that is internal to the IBM Storage Scale RAID servers.

IBM Storage Scale RAID features

This section introduces three key features of IBM Storage Scale RAID and how they work: data redundancy using RAID codes, end-to-end checksums, and declustering.

RAID codes

IBM Storage Scale RAID corrects for disk failures and other storage faults automatically by reconstructing the unreadable data by using the available data redundancy of a Reed-Solomon code or *N*-way replication. IBM Storage Scale RAID uses the reconstructed data to fulfill client operations. If disk fails, IBM Storage Scale RAID uses the reconstructed data to rebuild the data onto spare space. IBM Storage Scale RAID supports 2- and 3-fault-tolerant Reed-Solomon codes and 3-way and 4-way replication, which detects and corrects up to two or three concurrent faults[1]. The redundancy code layouts that IBM Storage Scale RAID supports, called *tracks*, are illustrated in Figure 1 on page 3.

[1] An f-fault-tolerant Reed-Solomon code or a $(1 + f)$-way replication can survive the concurrent failure of f disks or read faults. Also, if there are s equivalent spare disks in the array, an f-fault-tolerant array can survive the sequential failure of $f + s$ disks where disk failures occur between successful rebuild operations.

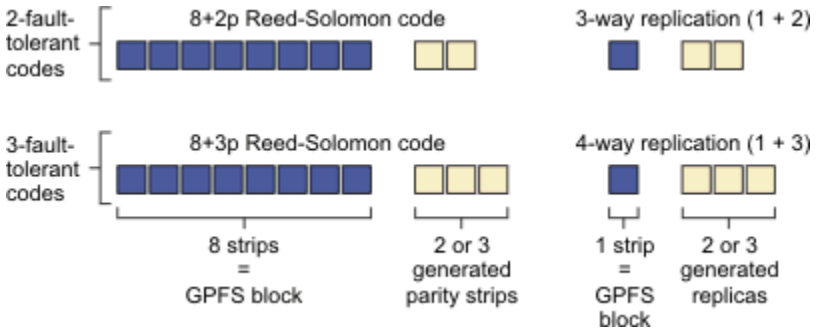


Figure 1. Redundancy codes supported by IBM Storage Scale RAID

Depending on the configured RAID code, IBM Storage Scale RAID creates redundancy information automatically. Using a Reed-Solomon code, IBM Storage Scale RAID divides a GPFS block of user data equally into eight data strips and generates 2 or 3 redundant parity strips. This results in a stripe or track width of 10 or 11 strips and storage efficiency of 80% or 73%, respectively (excluding user-configurable spare space for rebuild operations).

Using *N*-way replication, a GPFS data block is replicated simply *N* – 1 times, in effect implementing 1 + 2 and 1 + 3 redundancy codes, with the strip size equal to the GPFS block size. Thus, for every block/strip that is written to the disks, *N* replicas of that block/strip are also written. This results in a track width of 3 or 4 strips and storage efficiency of 33% or 25%, respectively.

End-to-end checksum

Most implementations of RAID codes implicitly assume that disks reliably detect and report faults, hard-read errors, and other integrity problems. However, studies show that disks do not report some read faults and occasionally fail to write data, while claiming to have written the data. These errors are often referred to as silent errors, phantom-writes, dropped-writes, and off-track writes. To cover for these shortcomings, IBM Storage Scale RAID implements an end-to-end checksum that can detect silent data corruption that is caused by either disks or other system components that transport or manipulate the data.

When an NSD client is writing data, a checksum of 8 bytes is calculated and appended to the data before it is transported over the network to the IBM Storage Scale RAID server. On reception, IBM Storage Scale RAID calculates and verifies the checksum. Then, IBM Storage Scale RAID stores the data, a checksum, and version number to disk and logs the version number in its metadata for future verification during read.

When IBM Storage Scale RAID reads disks to satisfy a client read operation, it compares the disk checksum against the disk data and the disk checksum version number against what is stored in its metadata. If the checksums and version numbers match, IBM Storage Scale RAID sends the data along with a checksum to the NSD client. If the checksum or version numbers are invalid, IBM Storage Scale RAID reconstructs the data by using parity or replication and returns the reconstructed data and a newly generated checksum to the client. Thus, both silent disk read errors and lost or missing disk writes are detected and corrected.

Declustered RAID

Compared to conventional RAID, IBM Storage Scale RAID implements a sophisticated data and spare space disk layout scheme that allows for arbitrarily sized disk arrays while also reducing the overhead to clients when recovering from disk failures. To accomplish this, IBM Storage Scale RAID uniformly spreads or *declusters* user data, redundancy information, and spare space across all the disks of a declustered array. Figure 2 on page 4 compares a conventional RAID layout versus an equivalent declustered array.

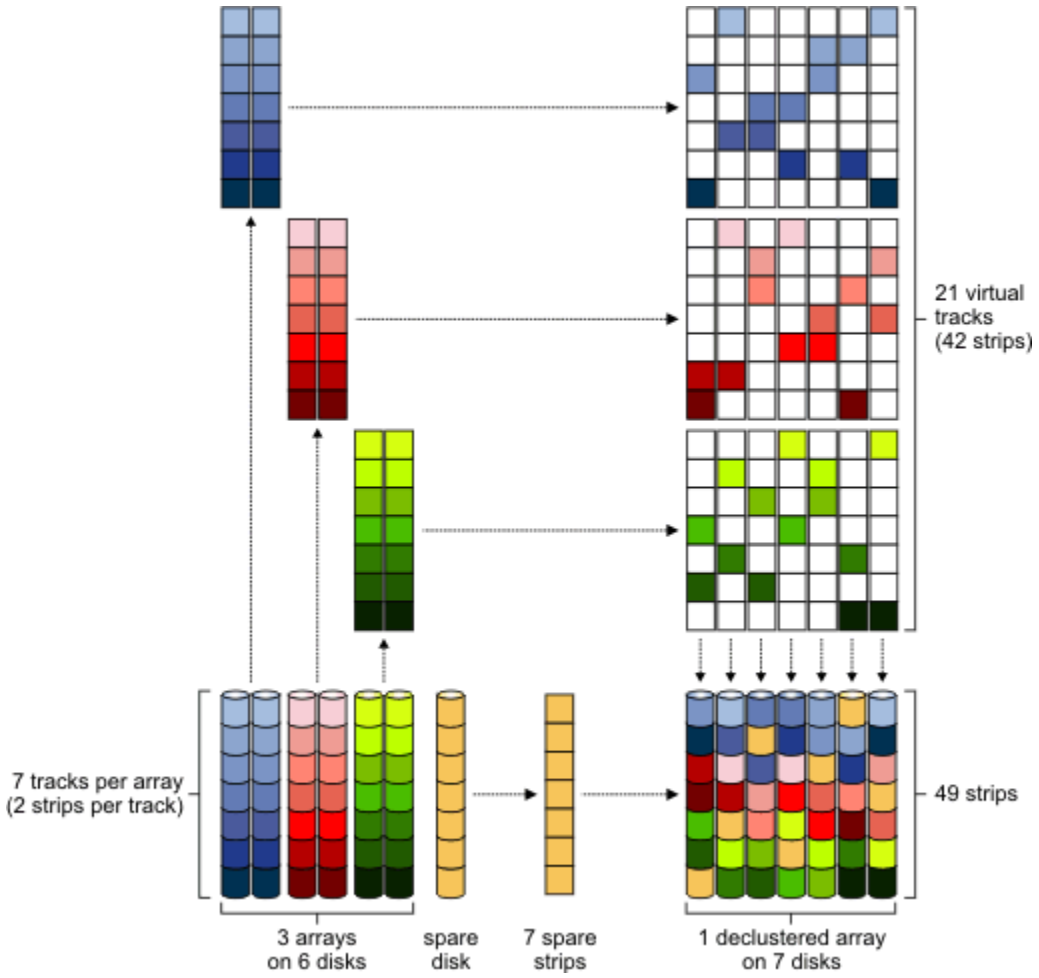


Figure 2. Conventional RAID versus declustered RAID layouts

As illustrated in Figure 3 on page 4, a declustered array can significantly shorten the time that is needed to recover from a disk failure, which lowers the rebuild overhead for client applications. When a disk fails, erased data is rebuilt by using all the operational disks in the declustered array, the bandwidth of which is greater than the fewer disks of a conventional RAID group. Furthermore, if an extra disk fault occurs during a rebuild, the number of impacted tracks that require repair is markedly less than the previous failure and less than the constant rebuild overhead of a conventional array.

The decrease in declustered rebuild impact and client overhead can be a factor of 3 to 4 times less than a conventional RAID. Because IBM Storage Scale stripes client data across all the storage nodes of a cluster, file system performance becomes less dependent upon the speed of any single rebuilding storage array.

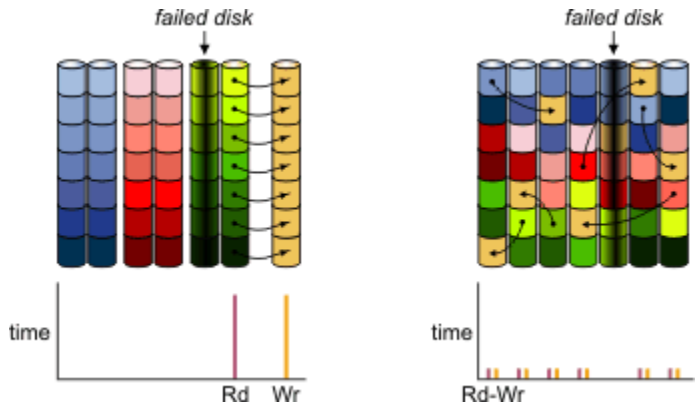


Figure 3. Lower rebuild overhead in declustered RAID versus conventional RAID

Disk configurations

This section describes recovery group and declustered array configurations.

Recovery groups

IBM Storage Scale RAID divides disks into *recovery groups* where each is physically connected to two servers: primary and backup. All accesses to any of the disks of a recovery group are made through the active server of the recovery group, either the primary or backup.

Building on the inherent NSD failover capabilities of IBM Storage Scale, when an IBM Storage Scale RAID server stops operating because of a hardware fault, software fault, or normal shutdown, the backup IBM Storage Scale RAID server seamlessly takes over control of the associated disks of its recovery groups.

Typically, a JBOD array is divided into two recovery groups that are controlled by different primary IBM Storage Scale RAID servers. If the primary server of a recovery group fails, control automatically switches over to its backup server. Within a typical JBOD, the primary server for a recovery group is the backup server for the other recovery group.

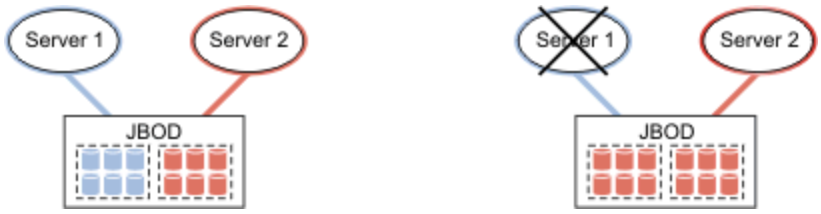


Figure 4. Minimal configuration of two IBM Storage Scale RAID servers and one storage JBOD

Declustered arrays

A declustered array is a subset of the physical disks (pdisks) in a recovery group across which data, redundancy information, and spare space are declustered. The number of disks in a declustered array is determined by the RAID code-width of the vdisks that is housed in the declustered array. For more information, see "Virtual disks" on page 6. There can be one or more declustered arrays per recovery group. Figure 5 on page 6 illustrates a storage JBOD with two recovery groups, each with four declustered arrays.

A declustered array can hold one or more vdisks. Since redundancy codes are associated with vdisks, a declustered array can simultaneously contain both Reed-Solomon and replicated vdisks.

If the storage JBOD supports multiple disks that are physically mounted together on removable carriers, removal of a carrier temporarily disables access to all the disks in the carrier. Thus, pdisks on the same carrier must not be in the same declustered array, as vdisk redundancy protection would be weakened upon carrier removal.

Declustered arrays are normally created at recovery group creation time but new ones can be created or existing ones that are grown by adding pdisks later.

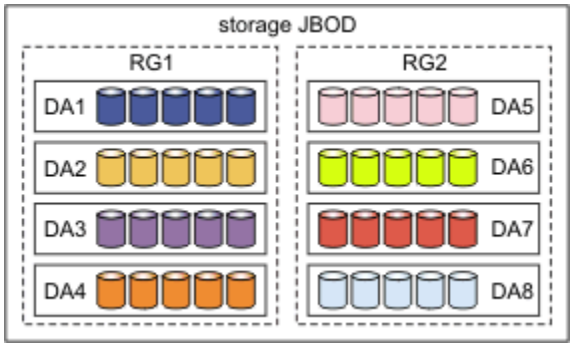


Figure 5. Example of declustered arrays and recovery groups in storage JBOD

Virtual and physical disks

A virtual disk (vdisk) is a type of NSD, implemented by IBM Storage Scale RAID across all the physical disks (pdisks) of a declustered array. Multiple vdisks can be defined within a declustered array, typically Reed-Solomon vdisks for GPFS user data and replicated vdisks for GPFS metadata.

Virtual disks

Whether a vdisk of a particular capacity can be created in a declustered array depends on its redundancy code, the number of pdisks and equivalent spare capacity in the array, and other small IBM Storage Scale RAID overhead factors. The **mmcrvdisk** command can automatically configure a vdisk of the largest possible size given a redundancy code and configured spare space of the declustered array.

In general, the number of pdisks in a declustered array cannot be less than the widest redundancy code of a vdisk plus the equivalent spare disk capacity of a declustered array. For example, a vdisk that uses the 11-strip-wide 8 + 3p Reed-Solomon code requires at least 13 pdisks in a declustered array with the equivalent spare space capacity of two disks. A vdisk that uses the 3-way replication code requires at least five pdisks in a declustered array with the equivalent spare capacity of two disks.

Vdisks are partitioned into virtual tracks, which are the functional equivalent of a GPFS block. All vdisk attributes are fixed at creation and cannot be altered later.

Physical disks

IBM Storage Scale RAID uses pdisks to store user data and IBM Storage Scale RAID internal configuration data.

A pdisk is a conventional, rotating magnetic-media hard disk drive (HDD) or a solid-state disk (SSD). All pdisks in a declustered array must have the same capacity.

It is assumed that pdisks are dual-ported. In this type of configuration, one or more paths are connected to the primary IBM Storage Scale RAID server and one or more paths are connected to the backup server. Typically, there are two redundant paths between an IBM Storage Scale RAID server and connected JBOD pdisks.

Solid-state disks

Early versions of ESS used several solid-state disks (SSDs) in each recovery group in order to redundantly log changes to its internal configuration and fast-write data in non-volatile memory, accessible from the primary or backup IBM Storage Scale RAID servers after server failure.

Later versions of ESS typically use NVRAM, with a single SSD per recovery group that is only used as a backup for the NVRAM in case of failure.

IBM Storage Scale RAID with pdisk-group fault tolerance

IBM Storage Scale RAID has a revised placement algorithm for distributing strips of the redundancy code. The revision can allow survival of larger units of concurrent disk failures than what was possible in previous versions of IBM Storage Scale RAID. The placement algorithm is aware of the hardware groupings of disks that are present in the system and attempts to separate individual strips of a redundancy code stripe across as many groups as possible.

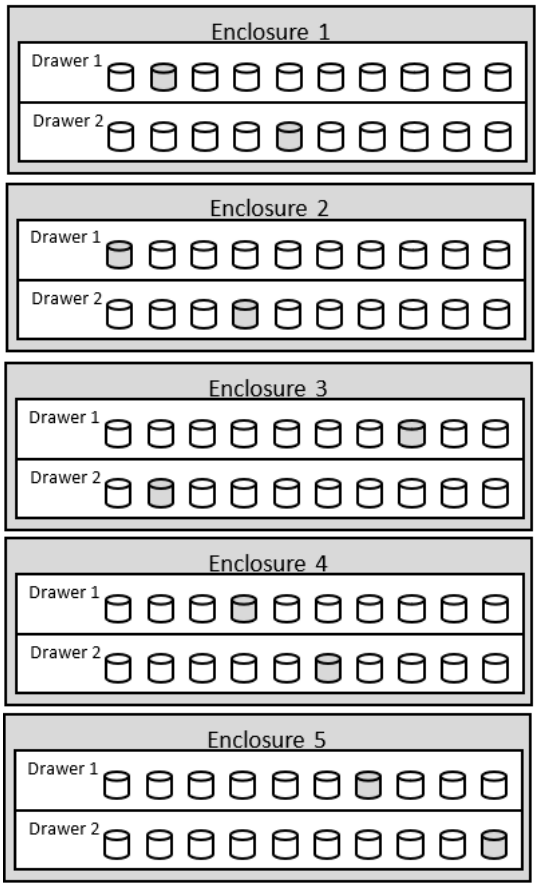
For example, if the hardware configuration includes four disk enclosures and a vdisk has been created with four-way replication, each strip of the vdisk's four-way stripe can be placed on a separate enclosure. Furthermore, if a complete enclosure (potentially many tens or hundreds of disks) were to fail, the surviving redundancy code strips on other enclosures would ensure no data loss. This revised placement is significantly different from the placement that is exhibited in previous versions of IBM Storage Scale RAID, which made no attempt to separate strips across hardware groupings of disks and thus might have placed all four redundancy code strips within one enclosure. The loss of that one enclosure would cause the data to be unavailable.

IBM Storage Scale RAID uses redundancy codes that are user who is selected for user data and system that is selected for configuration data. The selected redundancy code, available disk space, and current disk hardware configuration all play a role regarding types of failures can be survived. IBM Storage Scale RAID selects a minimum of five-way replication for its internal configuration data and requires a certain amount of physical disk space to be available for describing the system. IBM Storage Scale RAID also discovers the disk hardware groups automatically and uses this discovery in a periodic rebalance of the redundancy code strips. If the disk hardware configuration changes (if a new disk enclosure is added to the recovery group, for example), IBM Storage Scale RAID recognizes the change automatically and performs a rebalancing operation automatically in the background. Additionally, a rebuild operation when hardware failure is also cognizant of the hardware groupings, so failed redundancy code strips are rebuilt in a manner that is aware of the current disk hardware grouping.

Pdisk-group fault tolerance: an example

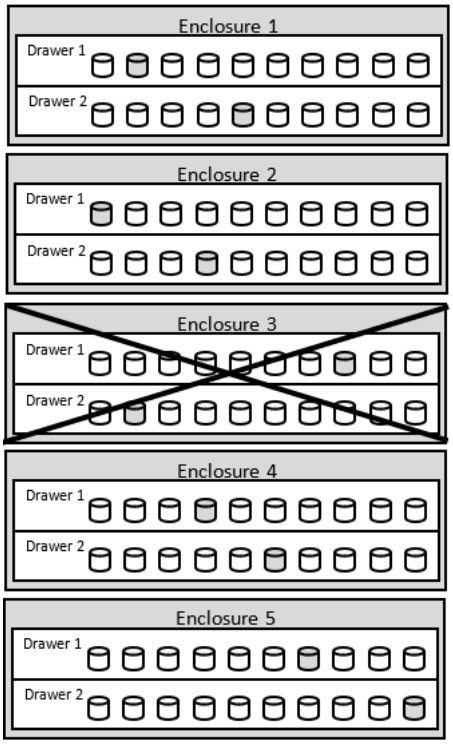
Every data stripe (including user data and system configuration data) within the IBM Storage Scale RAID system is protected through a distinct form of redundancy. Each of these data stripes has a set of disks within which they constrain their strip placement. Each stripe of the data (for which there are many stripes in each whole) has individual strips that serve in the redundancy code protection of the object's data. The placement of these strips is distributed across a set of pdisks residing within a set of drawers. These drawers reside within a set of enclosures.

Figure 1 shows a sample stripe placement for a vdisk that was using a RAID redundancy code of 8+2p (that is, eight data strips and two parity strips) on five enclosure system. The pdisk-group fault-tolerant placement has chosen to place the 10 strips of the stripe across five enclosures (each having two drawers).



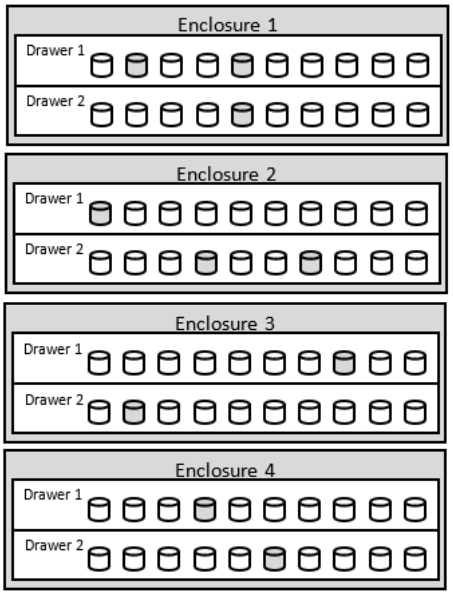
By segregating each individual strip across as wide a set of disk groups as possible, IBM Storage Scale RAID ensures that the loss of any set of disk groups up to fault tolerance of the RAID redundancy code is survivable. So in the give example the pdisk-group fault-tolerance is one enclosure because RAID code can survive two strips.

Figure 2 shows an example of the same configuration after the loss of a full enclosure. You can see that there are eight strips of the stripe that are still available, hence data will be fully available even after the failure of one enclosure.



After the failure, the GNR software tries to rebuild the DA subject to availability of space such that maximum fault tolerance can be achieved.

For the same 8+2p RAID code if you had four enclosure system, the pdisk-group fault-tolerance is one drawer. This RAID code under this configuration cannot survive an enclosure failure.



If you want enclosure failure on four enclosure system, you would need to set the RAID code to 8+3p.

Limiting factor of pdisk-group fault-tolerance

IBM Storage Scale RAID selects a minimum of five-way replication for its internal configuration data and requires some physical disks to be available for describing multiple entities within recovery group, internally called recovery group (rg) descriptor. Similarly, pdisk-group fault tolerance is also used for system vdisks like loghome, logtip, and logtip backup. The actual pdisk-group fault-tolerance is a union of distribution of actual strips of vdisk and internal recovery group descriptor across available failure

domains (nodes, enclosure, drawer, pdisks). The fault tolerance of internal configuration data is the limiting factor for any system or user vdisk. Hence in some cases actual pdisk-group fault-tolerance will be lower than theoretical pdisk-group fault tolerance.

Note: Always refer to command output to find actual pdisk-group fault-tolerance as it varies with RAID code and system configuration.

In the new version, pdisk-group fault-tolerance can be seen through **mmvdisk** command

```
mmvdisk recoverygroup list --recovery-group <RgName> –all
```

or

```
mmvdisk recoverygroup list --recovery-group <RgName> --fault-tolerance
```

In older version, this can be seen through the following command:

```
mmlsrecoverygroup <RgName> -L
```

The following example from six enclosure system showing system vdisk RG001LOGHOME pdisk-group fault-tolerance. Theoretically it should be three enclosures but it is limited by recovery group descriptor and hence the actual fault-tolerance is two enclosures .

```
configuration data  disk group fault tolerance         remarks
------------------  --------------------------------  -------
rg descriptor       2 enclosure                       limiting fault tolerance
system index        2 enclosure                       limited by rg descriptor

vdisk               RAID code        disk group fault tolerance         remarks
------------------  ---------------  --------------------------------  -------
RG001LOGHOME        4WayReplication  2 enclosure                       limited by rg descriptor
RG001LOGTIP         2WayReplication  1 pdisk
RG001LOGTIPBACKUP   Unreplicated     0 pdisk
RG001VS001          8+2p             1 enclosure
RG001VS002          8+2p             1 enclosure
```

The following example from two enclosure (without drawer) system shows system vdisk RG001LOGHOME and user vdisk RG001VS004 pdisk-group fault-tolerance that are limited by recovery group descriptor, which is lower than theoretical max.

```
configuration data  disk group fault tolerance         remarks
------------------  --------------------------------  -------
rg descriptor       4 pdisk                           limiting fault tolerance
system index        4 pdisk                           limited by rg descriptor

vdisk               RAID code        disk group fault tolerance         remarks
------------------  ---------------  --------------------------------  -------
RG001LOGHOME        4WayReplication  3 pdisk                           limited by rg descriptor
RG001LOGTIP         2WayReplication  1 pdisk
RG001LOGTIPBACKUP   Unreplicated     0 pdisk
RG001VS001          8+2p             2 pdisk
RG001VS004          3WayReplication  2 pdisk                           limited by rg descriptor
```

Disk hospital

The disk hospital is a key feature of IBM Storage Scale RAID that asynchronously diagnoses errors and faults in the storage subsystem. IBM Storage Scale RAID times out an individual pdisk I/O operation after about ten seconds, thereby limiting the impact from a faulty pdisk on a client I/O operation. When a pdisk I/O operation results in a timeout, an I/O error, or a checksum mismatch, the suspect pdisk is immediately admitted into the disk hospital. When a pdisk is first admitted, the hospital determines whether the error was caused by the pdisk itself or by the paths to it. While the hospital diagnoses the error, IBM Storage Scale RAID, if possible, uses vdisk redundancy codes to reconstruct lost or erased strips for I/O operations that would otherwise use the suspect pdisk.

Health metrics

The disk hospital maintains the following internal health assessment metrics for each pdisk. When one of these metrics exceeds the threshold, the pdisk is marked for replacement according to the disk maintenance replacement policy for the declustered array.

relativePerformance
Characterizes response times. Values greater than one indicate that the disk is performing above average speed; values less than one indicate that the disk is performing below average speed. Values within a range of 0.800 to 1.250 are considered normal. Examples of typical values are: 0.932, 0.978, 1.039, and 1.095. If the `relativePerformance` of a disk falls below a particular threshold, the hospital adds "slow" to the pdisk state and the disk is prepared for replacement. To check the current value, run the **`mmlsconfig nsdRAIDDiskPerformanceMinLimitPct`** command.

bit error rate
Characterizes media errors (hard errors) and checksum errors.

The disk hospital logs selected Self-Monitoring, Analysis and Reporting Technology (SMART) data, including the number of internal sector remapping events for each pdisk.

Pdisk discovery

IBM Storage Scale RAID discovers all connected pdisks when it starts up, and then regularly schedules a process that will rediscover a pdisk that becomes newly accessible to the IBM Storage Scale RAID server. This allows pdisks to be physically connected or connection problems to be repaired without restarting the IBM Storage Scale RAID server.

Disk replacement recording and reporting

The disk hospital keeps track of disks that require replacement according to the disk replacement policy of the declustered array, and it can be configured to report the need for replacement in various ways. It records and reports the FRU number and physical hardware location of failed disks to help guide service personnel to the correct location with replacement disks.

If the storage JBOD supports multiple disks that are mounted on a removable carrier, such as the Power 775, disk replacement requires the hospital to suspend other disks in the same carrier temporarily. On the Power 775 storage JBOD, the disk carriers are also not removable until IBM Storage Scale RAID actuates a solenoid-controlled latch to guard against human error.

In response to administrative commands, the hospital quiesces the appropriate disk (or multiple disks on a carrier), releases the carrier latch solenoid (if necessary), and turns on identify lights to guide replacement. After one or more disks are replaced and the disk or carrier is reinserted, the hospital, in response to administrative commands, verifies that the repair takes place and adds any new disks to the declustered array automatically, which causes IBM Storage Scale RAID to rebalance the tracks and spare space across all the disks of the declustered array. If service personnel fail to reinsert the disk or carrier within a reasonable period, the hospital declares the disks missing and starts rebuilding the affected data.

ESS fabric hospital

The ESS fabric hospital monitors hardware problems in a storage fabric. The fabric hospital identifies and isolates problems that impact the I/O availability.

The fabric hospital groups I/O error information based on hardware topology, and allows users to set thresholds when a group of errors exceeds a threshold. For example, all errors under a common SAS HBA port are grouped together and stored. If I/O errors on that system were isolated to the common port, then a corresponding event associated with the specified port is raised.

The ESS fabric hospital is only supported on the ESS 3500 and later models, and it supports to the SAS-based error monitoring.

Related tasks

"Installing and configuring the ESS fabric hospital" on page 12
The Zimon performance monitoring must be configured on the ESS fabric hospital to monitor events for the ESS building blocks management.

Installing and configuring the ESS fabric hospital

The Zimon performance monitoring must be configured on the ESS fabric hospital to monitor events for the ESS building blocks management.

Procedure

To manage ESS building blocks, configure the GUI setup on the EMS node.

```
essrun -N ems1,essio1,essio2 gui --configure
```

For more information, see https://www.ibm.com/docs/en/ess/6.1.6_ent?topic=guide-ess-deployment-quick-sheet.

Consequently, this command installs appropriate baseline packages and settings that are needed for the ESS fabric hospital setup.

Configuring performance monitoring tool sensors

The ESS fabric hospital uses custom performance monitoring tool sensors to relay path error information back to the performance monitoring tool collector node. By default, this node is the EMS node.

About this task

Due to current scaling concerns, it is recommended to install and configure only the performance monitoring tool pmsensors on 5-10 ESS building blocks. Where, each building block is an ESS server pair.

Procedure

1. Identify building blocks to install the sensors on them. The blocks are identified by passing a list of ESS nodes or a node class. This node class contains the nodes that will be monitored.
2. Run the following command from the EMS or I/O node for each recovery group definition for all building blocks that are being monitored:

   This step is necessary to prevent spurious RAS events and discard data that existed before the performance monitoring tool sensor is being installed.

   ```
   RGNAMES=rg1 rg2 rg3
   for rg in $RGNAMES
   do
     tsgnrgethospdata $rg  --reap --all-node-path
   done
   sleep 151
   mmperfmon config add --sensors /opt/IBM/zimon/defaults/ZIMonSensors_GPFSFabricHospital.cfg
   ```

3. Instead of using the code block in Step 2, it is recommended to use a sample script that is provided in `/usr/lpp/mmfs/samples/vdisk/install_essfabrichospital_sensor`.

   ```
   install_essfabrichospital_sensor [-h] [-f FILE][--override-sleep OVERRIDE_SLEEP]
   node_class_list
   ```

   where,

   node_class_list
   Specifies node class list (one per building block), which is separated by commas.

The node_class_list argument is different from the restrict argument that is provided in the /opt/IBM/zimon/defaults/ZIMonSensors_GPFSFabricHospital.cfg file and it is used to determine a sleeping period to synchronize the sensors.

-h, --help
Shows the help message and exits.

-f FILE, --file FILE
Specifies the sensor configuration file (default /opt/IBM/zimon/defaults/ZIMonSensors_GPFSFabricHospital.cfg).

--override-sleep OVERRIDE_SLEEP
Overrides the sleep delay when you are installing a sensor (debug).

The provided sensor file in /opt/IBM/zimon/defaults/ZIMonSensors_GPFSFabricHospital.cfg has the following contents:

```
sensors = {
        name = "GPFSFabricHospital"
        # This sensor should be activated only when ECE/GNR is configured
        # Only supported value is 900 seconds (15 minutes)
        period = 900
        restrict = "nsdNodes"
        type = "Generic"
}
```

Important: Only values of 900 (15 minutes in seconds) are supported.

4. Verify the new sensor.

```
# mmperfmon config show | grep GPFSFabricHospital -A3
```

A sample output is as follows:

```
name = "GPFSFabricHospital"
        period = 900
        restrict = "FabricHospital10BB"
        type = "Generic"
```

Note: In its current form, it is recommended to limit the sensor configuration to 10 ESS building block node pairs. Here, the system provided node class is "FabricHospital10BB", which contains all ESS server nodes by default. If your cluster has more than 10 ESS building block node pairs, then a custom node class should be defined that encapsulates up to 10 node pairs.

For example, assume that the "FabricHospital10BB" node class is created to contain the nodes that you want to monitor. The performance monitoring tool sensor file would be modified as follows:

```
# mmperfmon config show | grep GPFSFabricHospital -A3
```

A sample output is as follows:

```
 name = "GPFSFabricHospital"
 period = 900
 restrict = "FabricHospital10BB"
 type = "Generic"
```

5. Ensure that the performance monitoring component is healthy.

```
# mmhealth node show perfmon
```

A sample output is as follows:

```
Node name:      c145f11san06a-ib0.gpfs.net
Component     Status        Status Change     Reasons & Notices
------------------------------------------------------------------------------
PERFMON       HEALTHY       1 day ago         -
```

Note: Sensors can be configured only once. If you want to reinstall the sensor from scratch to change its behavior, you can run the following command to remove the sensor so that the sensor can be added again:

```
# mmperfmon config delete {--all |--sensors  Sensor[,Sensor...] }
```

Adding threshold rules

Add threshold rules after configuring the performance monitoring tool sensor.

Procedure

1. Restart the `mmsysmon` component.

   ```
   mmsysmoncontrol restart
   ```

2. Issue the following commands to install threshold rules:

   ```
   mmhealth thresholds add gpfs_fabhospital_errorIOCount:min
                          --groupBy
   node,gpfs_fabhospital_adapter,gpfs_fabhospital_port,gpfs_fabhospital_enclosure_id
                          --filterBy 'gpfs_fabhospital_version=3200' --warnlevel 5 --errorlevel 10
                          --sensitivity 900 --name SASPortErrorThreshold
   ```

   ```
   mmhealth thresholds add gpfs_fabhospital_errorIOCount:min
                         --groupBy gpfs_fabhospital_enclosure_id --filterBy
   'node=.*,gpfs_fabhospital_version=3200'
                         --warnlevel 5 --errorlevel 10 --sensitivity 900 --name
   SASEnclosureErrorThreshold
   ```

   The sensitivity is the time window for tracking a threshold. The warning levels, error levels, and sensitivity values may be changed away from the provided defaults.

   Note:

   - You cannot add multiple time-based thresholds to the same component. To change thresholds, you must first delete the previous setting by using the **`mmhealth thresholds delete`** command.
   - Thresholds can only be applied after data has been collected, which might require minimum 900 seconds waiting time. If you are using the provided sample `/usr/lpp/mmfs/samples/vdisk/install_essfabrichospital_sensor`, then no additional waiting time is needed.

Identifying threshold events

You can identify threshold events by referring an output example.

Procedure

1. If an event is triggered for one of the thresholds, and the event is currently active, it can be viewed by issuing the following command:

   ```
   # mmhealth node show threshold
   ```

   The SASPortErrorThreshold and the SASEnclosureErrorThreshold warning events are displayed in the following output example:

```
Node name:      c145f11san06a-ib0.gpfs.net

Component                      Status        Status Change     Reasons & Notices
-------------------------------------------------------------------------------------------------------------------
THRESHOLD                      DEGRADED      14 min. ago       thresholds_warn(SASEnclosureErrorThreshold,
                                                               SASPortErrorThreshold, SASPortErrorThreshold)
  MemFree_Rule                 HEALTHY       1 day ago         -
  SASEnclosureErrorThreshold   DEGRADED      14 min. ago       thresholds_warn(SASEnclosureErrorThreshold)
  SASPortErrorThreshold        DEGRADED      4 min. ago        thresholds_warn(SASPortErrorThreshold, SASPortErrorThreshold)
  active_thresh_monitor        HEALTHY       1 day ago         -
```

```
Event            Parameter                  Severity    Active Since  Event Message
---------------------------------------------------------------------------------------------------------------
thresholds_warn  SASEnclosureErrorThreshold  WARNING     14 min. ago   The value of gpfs_fabhospital_errorIOCount for
                                                                       the component(s) SASEnclosureErrorThreshold/
                                                                       (c145f11san06a-ib0.gpfs.net/3200/4/1
                                                                       /CFNH032/49,c145f11san06b-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/12)
                                                                       exceeded the threshold warning level 5 defined
                                                                       in SASEnclosureErrorThreshold.
thresholds_warn  SASPortErrorThreshold       WARNING     4 min. ago    The value of gpfs_fabhospital_errorIOCount for
                                                                        the component(s) SASPortErrorThreshold/
                                                                       (c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/49,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/94,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/83,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/52,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/55,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/85,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/46,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/13,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/88,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/93,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/10,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/6,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/11,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/7,
                                                                       c145f11san06a-ib0.gpfs.net/3101/4/1/CFNH032/92,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/5,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/91,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/70,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/98,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/86,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/71,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/3,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/12,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/53,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/97,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/96,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/102,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/45,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/15,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/48,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/99,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/82,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/4,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/2,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/84,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/87,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/47,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/44,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/14,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/100,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/56,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/8,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/51,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/90,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/81,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/89,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/9,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/1,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/95,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/101,
                                                                       c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/54)
                                                                       exceeded the threshold warning level 5 defined
                                                                       in SASPortErrorThreshold.
thresholds_warn  SASPortErrorThreshold       WARNING     4 min. ago    The value of gpfs_fabhospital_errorIOCount for the
                                                                       component(s) SASPortErrorThreshold/(c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/5,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/91,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/12,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/49,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/94,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/51,c145f11san06a-ib0. gpfs.net
                                                                       /3200/4/2/CFNH032/13,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/48,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/98,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/81,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/89,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/8,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/93,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/96,c145f11san06a-ib0.gpfs.net
                                                                       /3101/4/2/CFNH032/83,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/7,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/97,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/53,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/2,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/87,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/99,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/11,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/86,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/92,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/82,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/45,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/46,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/102,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/84,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/101,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/15,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/100,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/14,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/52,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/55,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/71,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/1,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/95,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/70,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/88,c145f11san06a-ib0.gpfs.net
                                                                       /3200/4/2/CFNH032/90,c145f11san06a-ib0.gpfs.net
```

```
                                                    /3200/4/2/CFNH032/10,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/6,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/47,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/3,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/85,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/4,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/56,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/9,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/54,c145f11san06a-ib0.gpfs.net
                                                    /3200/4/2/CFNH032/44) exceeded the threshold warning
                                                    level 5 defined in SASPortErrorThreshold.
```

2. Identify SASEnclosureErrorThreshold events.

   SASEnclosureErrorThreshold events represent a joint event that is causing I/O disruption across an entire enclosure. When a SASEnclosureErrorThreshold event is raised, it can be identified as follows:

   a) Go to the Event Message column in the output example.

   b) In the first component that is listed in the output example, identify the node name component and the enclosure serial number from the components key. All components in this group have the same enclosure serial number and node.

3. Identify SASPortErrorThreshold events.

   SASPortErrorThreshold events represent a problem in a given connection point between a node and an enclosure (such as a SAS cable) or a connection point between enclosures (daisy-chain case only). When a SASPortErrorThreshold event is raised, identify it as follows:

   a) Go to the Event Message column in the output example.

   b) In the first component that is listed in the output example, identify the node name and the adapter number, and the adapter port pairing. All components in this group have the same node, adapter name, and adapter number.

   ESS fabric hospital data points are encoded as a topological key of the form:

```
c145f11san06a-ib0.gpfs.net/3200/4/1/CFNH032/49 -> <FQDN GPFS admin node name>.<key
version>.<Adapter name>.<Adapter port number>.<Enclosure serial>.<Enclosure slot number>.
```

   Notes:

   - <Enclosure serial> and <Enclosure slot number> are the basis of the GNR pdisk location code, which takes the <Enclosure serial>-<Enclosure slot number> form in most cases.
   - SASPortErrorThreshold and SASEnclosureErrorThreshold are not mutually exclusive: when an enclosure event is raised, it is likely that SASPortErrorThreshold will be raised on the connections that overlap the enclosure. In this case, ignore the SASPortErrorThreshold event and consider the SASEnclosureErrorThreshold as the higher-level error.
   - If you are using a daisy-chained enclosure configuration, pay attention to the enclosures represented in the output list for SASPortErrorThreshold . If all enclosures are not reachable by the adapter port pair are represented in the event, then it might be that the enclosure interlink is malfunctioning, as opposed to the problem source being the upstream host adapter port.

Disabling the ESS fabric hospital

To disable the ESS fabric hospital, this procedure is recommended.

Procedure

1. Remove the threshold rules that were added during the ESS hospital configuration.

```
# mmhealth thresholds delete {RuleName | all}
```

2. To prevent additional data that is being collected, remove the performance monitoring tool sensor.

```
# mmperfmon config delete --sensors  GPFSFabricHospital
```

Chapter 2. Administering IBM Storage Scale RAID

Before you perform IBM Storage Scale RAID administration tasks, review information about getting started with IBM Storage Scale, requirements for administering IBM Storage Scale RAID, and common command principles.

For information about getting started with IBM Storage Scale, see the *IBM Storage Scale: Concepts, Planning, and Installation Guide*.

For information about the administration and maintenance of IBM Storage Scale and your GPFS file systems, see the *IBM Storage Scale: Administration Guide*.

Requirements for administering IBM Storage Scale RAID

Root authority is required to perform all IBM Storage Scale RAID administration tasks.

IBM Storage Scale RAID commands maintain the appropriate environment across all nodes in the GPFS cluster. To achieve this, IBM Storage Scale RAID commands use the remote shell and remote file copy commands that you specify on the `mmcrcluster` command or the `mmchcluster` command. See the *IBM Storage Scale: Command and Programming Reference* for more information.

The default remote commands are `rsh` and `rcp`, but you can designate `ssh` and `scp` or any other remote commands with compatible syntax. The `rsh` and `rcp` commands that are provided by the Windows Cygwin environment do not support IBM Storage Scale. If your cluster includes Windows nodes, you must designate `ssh` and `scp` as the remote communication program.

In principle, you can issue IBM Storage Scale RAID administration commands from any node in the GPFS cluster. The nodes that you plan to use for administering IBM Storage Scale RAID must be able to run remote shell commands on themselves and on any other node in the cluster without the use of a password and without producing any extraneous messages. Similarly, the nodes on which the IBM Storage Scale RAID commands are issued must be able to copy files to and from any other node in the GPFS cluster without the use of a password and without producing any extraneous messages.

The way the passwordless access is achieved depends on the particular remote execution program and authentication mechanism being used. For example, for `rsh` and `rcp`, you might need a properly configured `.rhosts` file in the root user's home directory on each node in the GPFS cluster. If the remote program is `ssh`, you can use private identity files that do not have a password. Or, if the identity file is password protected, you can use the `ssh-agent` utility to establish an authorized session before you issue `mm` commands. It is the administrator's responsibility to issue `mm` commands only from nodes that are configured properly and that can access the rest of the nodes in the GPFS cluster.

Common IBM Storage Scale RAID command principles

There are some common principles that you should keep in mind when you are running IBM Storage Scale RAID commands.

These principles include:

- Unless otherwise noted, IBM Storage Scale RAID commands can be run from any node in the GPFS cluster. Exceptions are commands that are not supported in a particular operating system environment. Certain commands might additionally require the affected GPFS file system to be mounted.
- IBM Storage Scale supports the "no" prefix on all Boolean type long (or dash-dash) options.

Specifying nodes as input to IBM Storage Scale RAID commands

About this task

Many IBM Storage Scale RAID commands accept a node or multiple nodes as part of their input, using the `-N` flag. Nodes can be specified with IBM Storage Scale RAID commands in a variety of ways:

Node
A representation of an individual node, which can be any of these:

- Short GPFS administration node interface name.
- Long GPFS administration node interface name.
- Short GPFS daemon node interface name.
- Long GPFS daemon node interface name.
- IP address corresponding to the GPFS daemon node interface.
- GPFS node number.

Node - Node
A node range, indicated by specifying two node numbers separated by a hyphen (-), with the first node number being less than or equal to the second node number. For example, node range 3-8 specifies the nodes with node numbers 3, 4, 5, 6, 7, and 8.

NodeClass
A set of nodes that are grouped into system-defined node classes or user-defined node classes. The system-defined node classes that are known to IBM Storage Scale are:

all
All of the nodes in the GPFS cluster.

clientnodes
All nodes that do not participate in file system administration activities.

localhost
The node on which the command is running.

managernodes
All nodes in the pool of nodes from which file system managers and token managers are selected.

mount
For commands involving a file system, all of the local nodes on which the file system is mounted (nodes in remote clusters are always excluded, even when they mount the file system in question).

nonquorumnodes
All of the non-quorum nodes in the GPFS cluster.

nsdnodes
All of the NSD server nodes in the GPFS cluster.

quorumnodes
All of the quorum nodes in the GPFS cluster.

User-defined node classes are created with the `mmcrnodeclass` command. After a node class is created, it can be specified as an argument on commands that accept the `-N` *NodeClass* option. User-defined node classes are managed with the `mmchnodeclass`, `mmdelnodeclass`, and `mmlsnodeclass` commands. See the *IBM Storage Scale: Command and Programming Reference* for more information.

NodeFile
A file that contains a list of nodes. A node file can contain individual nodes or node ranges.

For commands operating on a file system, the stripe group manager node is always implicitly included in the node list. Not every IBM Storage Scale RAID command supports all of the node specification options described in this topic. To learn which kinds of node specifications are supported by a particular IBM

Storage Scale RAID command, see the relevant command description in Appendix B, “IBM Storage Scale RAID commands,” on page 263.

Stanza files

The input to a number of IBM Storage Scale RAID commands can be provided in a file organized in a stanza format.

About this task

A stanza is a series of whitespace-separated tokens that can span multiple lines. The beginning of a stanza is indicated by the presence of a stanza identifier as the first token on a line. Stanza identifiers consist of the % (percent sign) character, followed by a keyword, and ending with the : (colon) character. For example, %nsd: indicates the beginning of an NSD stanza.

A stanza identifier is followed by one or more stanza clauses describing different properties of the object. A stanza clause is defined as an *Attribute=value* pair.

Lines that start with the # (pound sign) character are considered comment lines and are ignored. Similarly, you can imbed inline comments following a stanza clause; all text after the # character is considered a comment.

The end of a stanza is indicated by one of the following:

- a line that represents the beginning of a new stanza
- a blank line
- a non-comment line that does not contain the = character

IBM Storage Scale recognizes a number of stanzas:

%nsd:
NSD stanza

%pdisk:
Physical disk stanza

%vdisk:
Virtual disk stanza

%da:
Declustered array stanza

%rg:
Recovery group stanza

The details are documented under the corresponding commands.

A stanza file can contain multiple types of stanzas. Commands that accept input in the form of stanza files expect the stanzas to be syntactically correct but will ignore stanzas that are not applicable to the particular command. Similarly, if a particular stanza clause has no meaning for a given command, it is ignored.

For backward compatibility, a stanza file may also contain traditional NSD descriptors, although their use is discouraged.

Here is what a stanza file may look like:

```
# Sample file containing two NSD stanzas

# Example for an NSD stanza with imbedded comments
%nsd:     nsd=DATA5     # my name for this NSD
    device=/dev/hdisk5  # device name on node k145n05
     usage=dataOnly
    # List of server nodes for this disk
    servers=k145n05,k145n06
    failureGroup=2
    pool=dataPoolA
```

```
# Example for a directly attached disk; most values are allowed to default
%nsd:     nsd=DATA6   device=/dev/hdisk6   failureGroup=3
```

Chapter 3. Managing IBM Storage Scale RAID with the mmvdisk command

The **mmvdisk** command is an integrated command suite and management methodology for IBM Storage Scale RAID. It provides a unified conceptual framework that simplifies many of the more complicated aspects of IBM Storage Scale RAID administration.

The **mmvdisk** command enforces and encourages IBM Storage Scale RAID best practices with respect to the following tasks:

- IBM Storage Scale RAID server configuration by using the **mmvdisk server** sub-command.
- Recovery group creation and management by using the **mmvdisk recoverygroup** sub-command.
- The definition, sizing, and creation of vdisk NSDs by using the **mmvdisk vdiskset** sub-command.
- The creation of vdisk-based file systems by using the **mmvdisk filesystem** sub-command.

Compatibility between **mmvdisk** and the legacy IBM Storage Scale RAID command set is strictly limited. For example, a recovery group that has been created using **mmvdisk** cannot be deleted using **mmdelrecoverygroup**.

The **mmvdisk** command can manage all three types of IBM Storage Scale RAID recovery groups:

1. The *paired recovery groups* of Elastic Storage Server, where a pair of servers divide a common set of disk enclosures into a pair of recovery groups, with each server taking exclusive primary responsibility for one recovery group of the pair.
2. The *scale-out recovery groups* of IBM Storage Scale Erasure Code Edition, where a set of 4 to 32 identically equipped servers each take an equal share of responsibility in a single collective recovery group.
3. The *shared recovery groups* of IBM Elastic Storage System 3000, where a single recovery group is defined on a common disk enclosure shared by two servers.

A recovery group that is managed by the **mmvdisk** command is called an mmvdisk recovery group. It cannot be managed using the legacy IBM Storage Scale RAID command set.

A recovery group that is managed using the legacy IBM Storage Scale RAID command set is called either a non-mmvdisk recovery group or a legacy recovery group. It cannot be managed using **mmvdisk** other than to be converted from legacy management to mmvdisk management.

A new IBM Storage Scale cluster can use **mmvdisk** from the beginning to manage any IBM Storage Scale RAID recovery groups. The server and recovery group pairs of existing Elastic Storage Server installations can be converted to enable mmvdisk management. All existing non-mmvdisk recovery groups in a cluster must be converted to mmvdisk management before **mmvdisk** can be used to create new recovery groups in the cluster.

The scale-out recovery groups of IBM Storage Scale Erasure Code Edition and the shared recovery groups of IBM Elastic Storage System 3000 must be managed by **mmvdisk**. If a cluster containing non-mmvdisk paired recovery groups wishes to add scale-out or shared recovery groups, the existing paired recovery groups must first be converted to mmvdisk management.

When all recovery groups in an IBM Storage Scale cluster are managed by **mmvdisk**, the legacy IBM Storage Scale RAID commands will refuse to create new recovery groups. All further IBM Storage Scale RAID administration must be performed using **mmvdisk**.

For the management of IBM Storage Scale file system vdisk NSDs in recovery groups, the central concept in **mmvdisk** is that of a vdisk set. A vdisk set is a collection of uniform vdisk NSDs from one or more recovery groups. The member vdisk NSDs of a vdisk set all have the same attributes in each of the vdisk set's recovery groups. Vdisk sets provide a scalable specification template, which are applicable across multiple recovery groups for sizing and creating uniform vdisk NSDs. The vdisk NSDs of a vdisk set are then managed as a unit.

A vdisk-based file system is constructed from one or more vdisk sets. Each vdisk set in a file system contributes all of its member vdisk NSDs to the file system. A vdisk set must belong in its entirety to exactly one file system, but multiple entire vdisk sets can be combined to construct a single vdisk-based file system.

The mmvdisk administration methodology

The **mmvdisk** command structures IBM Storage Scale RAID around a well-defined collection of objects: node classes, servers, recovery groups, vdisk sets, and file systems.

Node classes

An mmvdisk node class is a regular IBM Storage Scale node class with the restriction that it can only be altered by the **mmvdisk** command.

For the paired recovery groups of ESS, the mmvdisk node class contains the pair of servers responsible for a recovery group pair. The **mmvdisk** command maintains an association between the node class and each of the paired recovery groups.

For the scale-out recovery groups of IBM Storage Scale Erasure Code Edition, the mmvdisk node class contains from 4 to 32 identically equipped servers for a single scale-out recovery group. The **mmvdisk** command maintains an association between the node class and the scale-out recovery group.

For the shared recovery groups of IBM Elastic Storage System 3000, the mmvdisk node class contains the two servers that share the single recovery group. The **mmvdisk** command maintains an association between the node class and the shared recovery group.

There can be no overlap among mmvdisk node classes. A server cannot be in two mmvdisk node classes, and the servers in an mmvdisk node class must either be the primary and backup servers for two related paired recovery groups or must all serve the same scale-out or shared recovery group.

Servers

The servers within an mmvdisk node class are expected to be homogeneous. Each server is of the same type, with the same processor, memory, network, and storage capability. The **mmvdisk** command explicitly enforces that each server in an mmvdisk node class has the same total real memory and the same server disk topology. The server disk topology is the collection of disks available for IBM Storage Scale RAID on a server. For scale-out recovery group servers, it is the number and type of disks exclusive to the individual server. For shared recovery groups and paired recovery groups, it is the configuration of twin-tailed enclosures shared by the server pair.

A server can belong to only one mmvdisk node class. For scale-out and shared recovery groups, a server belongs only to one recovery group. For paired recovery groups, a server belongs to the two recovery groups of a related recovery group pair.

For all recovery group types, the **mmvdisk** command uses the mmvdisk node class to maintain the IBM Storage Scale RAID configuration settings for the recovery group servers. The **mmvdisk** command never uses cluster-wide settings for recovery group servers since different recovery groups can use different settings. If recovery group server settings are ever made independently of **mmvdisk**, the same practice of using the mmvdisk node class should be followed.

Recovery groups

The **mmvdisk** command manages the three types of IBM Storage Scale RAID recovery groups:

1. The paired recovery groups of Elastic Storage Server. The two recovery groups in a paired recovery group node class are also called a *recovery group pair*.
2. The scale-out recovery groups of IBM Storage Scale Erasure Code Edition.
3. The shared recovery groups of IBM Elastic Storage System 3000.

A scale-out recovery group is created by supplying **mmvdisk** with a recovery group name and an mmvdisk node class containing 4 to 32 servers with the same independent disk configuration.

A recovery group pair is created by supplying **mmvdisk** with two recovery group names and an mmvdisk node class containing exactly two servers that share the same twin-tailed enclosure configuration.

A shared recovery group is created by supplying **mmvdisk** with a single recovery group name and an mmvdisk node class containing exactly two servers that share the same twin-tailed enclosure.

The association between a recovery group and an mmvdisk node class is established when the recovery group or recovery group pair is created.

The disk devices from the servers become the recovery group's pdisks. The pdisks are sorted into declustered arrays. A declustered array is a collection of pdisks that all have the same capacity in bytes and the same hardware type (SSD devices, NVMe devices, or HDD devices with the same rotation rate). Each pdisk in a declustered array is expected to have the same indexing and performance characteristics.

The space within a declustered array is allocated to vdisks. Vdisks are declustered RAID logical units that are balanced across the pdisks of a declustered array. There are two types of vdisks: log vdisks, which are used for IBM Storage Scale RAID transaction logging; and user vdisks, which become the vdisk NSDs of IBM Storage Scale file systems.

A recovery group is sub-divided into log groups. A log group is a collection of vdisks that share a common RAID transaction log (a log home vdisk). In the case of a recovery group pair, there is just one log group per recovery group, and in this case a paired recovery group is equivalent to its one and only log group. Each of the two servers has exclusive primary responsibility for the vdisks of one log group. In the case of a scale-out or shared recovery group, the recovery group is sub-divided into equally sized log groups, and each server is responsible for the vdisks of two log groups. Log groups act to equitably sub-divide and balance vdisks among the servers of a recovery group or recovery group pair.

The **mmvdisk** command considers the log vdisks of a recovery group as an essential component of the recovery group, and the formatting of log vdisks is part of recovery group creation.

Vdisk sets

File system vdisk NSDs are managed collectively as members of vdisk sets. A *vdisk set* is a collection of identical vdisk NSDs from one or more recovery groups. Each log group of a recovery group included in a vdisk set contributes one member vdisk NSD. This means that when a vdisk set is defined using paired recovery groups, there will be one member vdisk NSD from each recovery group because a paired recovery group is equivalent to a single log group. When a vdisk set is defined using scale-out or shared recovery groups, there will be one member vdisk NSD from each log group of the recovery group, which is equivalent to two vdisk NSDs per server of the scale-out or shared recovery group.

A vdisk set must be managed as a unit: All member vdisk NSDs of a vdisk set must belong to the same file system. Defining a vdisk set across multiple recovery groups helps to ensure that a file system is balanced using identical vdisk NSDs from the recovery groups in the vdisk set.

Once recovery groups are created, the **mmvdisk vdiskset** command is used to define vdisk sets across the recovery groups. A vdisk set definition is a specification template that permits administrators to preview and evaluate how the vdisk sets are sized within the servers and declustered arrays of the recovery groups.

When the vdisk sets have been defined and sized satisfactorily in the desired recovery groups, **mmvdisk** is used to create the vdisk sets. This instantiates a vdisk set definition into a real collection of vdisk NSDs.

The created vdisk sets are then used as the units from which **mmvdisk** builds IBM Storage Scale file systems.

File systems

An mmvdisk file system is an IBM Storage Scale file system where all of the vdisk NSDs in the file system are from vdisk sets.

It is possible for an mmvdisk file system to contain non-vdisk NSDs, but the vdisk NSDs must all be members of vdisk sets, and non-vdisk NSDs and vdisk NSDs cannot reside in the same file system storage pool.

It is also possible for a file system to contain some vdisk NSDs that are from vdisk sets and some vdisk NSDs that are not from vdisk sets. In this case, the file system is not an mmvdisk file system. A cluster could be in the process of being converted to mmvdisk administration, so that some of the vdisk NSDs are from converted recovery groups and some are not. The file system will not be an mmvdisk file system until the last legacy recovery group represented in the file system is converted to mmvdisk administration.

Overview of the mmvdisk commands

This topic provides an overview of the **mmvdisk** commands.

For comparison, each **mmvdisk** command is listed with the legacy IBM Storage Scale RAID commands that perform similar functions. For a complete description of the **mmvdisk** commands, see "mmvdisk command" on page 341.

mmvdisk nodeclass

The **mmvdisk nodeclass** command has the following primary functions:

1. To create the mmvdisk node class for the two servers of a recovery group pair.
2. To create the mmvdisk node class for the 4 to 32 servers of a scale-out recovery group.
3. To create the mmvdisk node class for the two servers of a shared recovery group.
4. To list mmvdisk node classes.

The **mmvdisk nodeclass** command performs functions similar to the **mmcrnodeclass**, **mmlsnodeclass**, **mmchnodeclass**, and **mmdelnodeclass** commands.

mmvdisk server

The **mmvdisk server** command has the following primary functions:

1. To list and validate the supported server disk topology of an mmvdisk node class.
2. To configure an mmvdisk node class to be IBM Storage Scale RAID recovery group servers.
3. To configure an individual server as preparation for being added to a scale-out recovery group.
4. To list the state of recovery group servers.

The **mmvdisk server** command performs functions similar to the **gssServerConfig**, **mmchconfig**, **mmlsconfig**, **mmshutdown**, **mmstartup**, **mmgetpdisktopology**, and **topsummary** commands.

mmvdisk recoverygroup

The **mmvdisk recoverygroup** command has the following primary functions:

1. To create an ESS recovery group pair on a configured mmvdisk node class.
2. To create an IBM Storage Scale Erasure Code Edition scale-out recovery group on a configured mmvdisk node class.
3. To create an IBM Elastic Storage System 3000 shared recovery group on a configured mmvdisk node class.
4. To list recovery group information.
5. To convert existing non-mmvdisk recovery group pairs to mmvdisk management.
6. To add, delete, or replace servers in a scale-out recovery group.
7. To perform supported MES upgrades on ESS recovery group pairs.

The **mmvdisk recoverygroup** command performs functions similar to the **mmgetpdisktopology**, **mkrginput**, **mmcrrecoverygroup**, **mmcrvdisk**, **mmlsrecoverygroup**, **mmchrecoverygroup**, **mmdelvdisk**, **mmaddpdisk**, **mmdelpdisk**, and **mmdelrecoverygroup** commands.

mmvdisk vdiskset

The **mmvdisk vdiskset** command has the following primary functions:

1. To define and size vdisk sets across recovery groups.
2. To create the member vdisk NSDs of a vdisk set according to its definition.
3. To list vdisk set attribute and sizing information.

The **mmvdisk vdiskset** command performs functions similar to the **mmcrvdisk**, **mmdelvdisk**, and **gssgenvdisks** commands.

mmvdisk filesystem

The **mmvdisk filesystem** has the following primary functions:

1. To create an mmvdisk file system using vdisk sets.
2. To add vdisk sets to mmvdisk file systems.
3. To delete vdisk sets from mmvdisk file systems.
4. To list mmvdisk file system information.

The **mmvdisk filesystem** command performs functions similar to the **mmcrfs**, **mmadddisk**, **mmdeldisk**, **mmdelfs**, **mmlsdisk**, and **mmchdisk** commands.

mmvdisk pdisk

The **mmvdisk pdisk** has the following primary functions:

1. To list recovery group pdisk information.
2. To list pdisks that are out of service or in need of replacement.
3. To replace pdisks that are in need of replacement.

The **mmvdisk pdisk** command performs functions similar to the **mmlspdisk**, **mmchcarrier**, and **mmchpdisk** commands.

mmvdisk vdisk

The primary function of the **mmvdisk vdisk** command is to list basic information about individual vdisks.

All other vdisk management is performed through the **mmvdisk vdiskset** command for file system vdisk NSDs, and through the **mmvdisk recoverygroup** command for recovery group log vdisks.

The **mmvdisk vdisk** command performs functions similar to the **mmlsvdisk** command.

Example of the mmvdisk command sequence

The **mmvdisk** command assumes that the IBM Storage Scale cluster is installed and configured properly with respect to licenses, quorum and node designations, and network connectivity. The intended IBM Storage Scale RAID recovery group server nodes must be members of the cluster.

In the case of ESS, the two server nodes should be properly connected to the enclosures that will provide the disks for the recovery group pair. In the case of IBM Storage Scale Erasure Code Edition, the intended server nodes should each be populated with the correct number and type of disks that will make up the scale-out recovery group. In the case of IBM Elastic Storage System 3000, the two server nodes should be properly connected to the enclosure that provides the disks for the shared recovery group.

The path to creating vdisk-based file systems with **mmvdisk** follows this sequence of steps:

1. Creating the mmvdisk node class. For a shared recovery group or for a recovery group pair, **mmvdisk** is used to create an mmvdisk node class for the server pair.

   ```
   # mmvdisk nodeclass create --node-class ESS01 -N server01,server02
   ```

   For a scale-out recovery group, the mmvdisk node class is created using all of the intended server nodes (from 4 to 32):

   ```
   # mmvdisk nodeclass create --node-class NC01 -N server01,server02,server03,server04,...
   ```

2. Verifying recovery group server disk topologies. Next, **mmvdisk** is used to verify that the intended supported server disk topology is recognized on the node class.

   ```
   # mmvdisk server list --node-class ESS01 --disk-topology
   ```

   Any errors in the disk topology should be recognized and corrected at this step.

3. Configuring recovery group servers. **mmvdisk** is then used to configure the mmvdisk node class as recovery group servers.

   ```
   # mmvdisk server configure --node-class ESS01 --recycle one
   ```

4. Creating recovery groups. For paired recovery groups, **mmvdisk** uses the mmvdisk node class to create the recovery group pair.

   ```
   # mmvdisk recoverygroup create --recovery-group ESS01L,ESS01R --node-class ESS01
   ```

   For a shared or scale-out recovery group, **mmvdisk** uses the mmvdisk node class to create the single recovery group.

   ```
   # mmvdisk recoverygroup create --recovery-group RG01 --node-class NC01
   ```

5. Defining vdisk sets. With the recovery group or recovery group pair created, **mmvdisk** is used to define vdisk sets. The vdisk set definitions are specification templates that are sized within one or more recovery groups, which allows the disk space and server memory demands to be previewed and evaluated before any vdisk NSDs are actually created. For a recovery group pair, the vdisk sets should be defined in both recovery groups.

   ```
   # mmvdisk vdiskset define --vdisk-set vs1 --recovery-group ESS01L,ESS01R --code 8+3p --block-size 4m --set-size 50%
   # mmvdisk vdiskset define --vdisk-set vs2 --recovery-group ESS01L,ESS01R --code 4+2p --block-size 512k --set-size 25%
   ```

   For a shared or scale-out recovery group, define the vdisk sets in the single recovery group.

   ```
   # mmvdisk vdiskset define --vdisk-set vs1 --recovery-group RG01 --code 8+3p --block-size 4m --set-size 50%
   # mmvdisk vdiskset define --vdisk-set vs2 --recovery-group RG01 --code 4+2p --block-size 512k --set-size 25%
   ```

   As long as the actual vdisk NSDs have not been created, vdisk sets may be undefined and defined again until the attributes and disk and memory sizings are satisfactory.

6. Creating vdisk NSDs. When the vdisk set definitions have been satisfactorily sized, **mmvdisk** is used to create the actual vdisk NSDs.

   ```
   # mmvdisk vdiskset create --vdisk-set vs1,vs2
   ```

7. Creating vdisk-based file systems. Once vdisk sets are created, **mmvdisk** is used to create file systems that use the vdisk NSDs from one or more vdisk sets.

   ```
   # mmvdisk filesystem create --file-system fs1 --vdisk-set vs1
   # mmvdisk filesystem create --file-system fs2 --vdisk-set vs2
   ```

These are the seven basic steps that are used to create mmvdisk file systems.

Node class management

Each mmvdisk recovery group must be associated with an mmvdisk node class that contains only the servers for the recovery group.

In the case of paired recovery groups, each recovery group in the pair will be associated with the same node class since it contains the primary and backup servers for that paired recovery group.

The **mmvdisk nodeclass create** command is used to create an mmvdisk node class that contains the intended servers for a recovery group or recovery group pair.

The node class is used by the **mmvdisk server configure** command to configure IBM Storage Scale RAID on the recovery group servers.

The association between an mmvdisk node class and an mmvdisk recovery group can be established in two ways:

1. The **mmvdisk nodeclass create** command is used to create a node class containing the intended servers for the recovery group, and the **mmvdisk recoverygroup create** command is used to create an mmvdisk recovery group using the node class.
2. The **mmvdisk recoverygroup convert** command is used to convert two paired recovery groups and their two servers to mmvdisk management. The **mmvdisk recoverygroup convert** command must be supplied with a node class name to be used as the mmvdisk node class for the converted recovery group pair.

Once a recovery group is created using an mmvdisk node class, the members of the node class can only be changed using the **mmvdisk recoverygroup** command.

When mmvdisk file systems span multiple recovery groups or recovery group pairs, each recovery group or recovery group pair is a single point of failure for its vdisk NSDs. In the case of a shared or scale-out recovery group, it is simply that the recovery group must be available for its vdisk NSDs to be available. In the case of a recovery group pair, the disk enclosures twin-tailed between the two servers are the single point of failure since each enclosure is split between the two recovery groups and will hold vdisk NSD data for both of the paired recovery groups. A shared recovery group can also be characterized as a single point of failure because of its twin-tailed enclosure.

The mmvdisk node class captures this relationship perfectly and is used by the **mmvdisk filesystem** command to automatically assign the failure groups for vdisk NSDs when a file system spans multiple recovery groups or recovery group pairs. Each vdisk NSD from the same node class is assigned the same file system failure group. If custom failure group numbers are desired, the **mmvdisk filesystem** command permits them to be assigned using the mmvdisk node class names as identifiers for the single point of failure.

The mmvdisk node class behaves in all other respects like a regular IBM Storage Scale node class.

Server management

The **mmvdisk** command considers a node to be a server if the node is a member of an mmvdisk node class, or if the node has a non-zero nsdRAIDTracks configuration setting.

Before a recovery group can be created, the servers in a node class should be verified to have the same memory and the same expected disk topology, and they must be configured to enable IBM Storage Scale RAID.

The memory configuration of a server node class can be checked using the **mmvdisk server list --config** command:

```
# mmvdisk server list --node-class ESS01 --config

 node
number  server                          active   memory    pagepool  nsdRAIDTracks
------  ------------------------------  -------  --------  --------  -------------
```

```
     1  ess01io1.gpfs.net                yes       123 GiB  1024 MiB             0
     2  ess01io2.gpfs.net                yes       123 GiB  1024 MiB             0
```

The disk topology of a server node class can be checked using the **mmvdisk server list --disk-topology** command:

```
# mmvdisk server list --node-class ESS01 --disk-topology

 node                                    needs    matching
number  server                          attention   metric   disk topology
------  ------------------------------  ---------  --------  -------------
     1  ess01io1.gpfs.net               no          100/100  ESS GL6
     2  ess01io2.gpfs.net               no          100/100  ESS GL6
```

Note: Recognition of server disk topologies depends on the appropriate ESS, IBM Storage Scale Erasure Code Edition, or IBM Elastic Storage System 3000 support RPMs being installed on all server nodes and on any node where the **mmvdisk** command is run.

The node class is configured using the **mmvdisk server configure** command:

```
# mmvdisk server configure --node-class ESS01 --recycle one
mmvdisk: Checking resources for specified nodes.
mmvdisk: Node 'ess01io1.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node 'ess01io2.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node class 'ESS01' has a paired recovery group disk topology.
mmvdisk: Setting configuration for node class 'ESS01'.
mmvdisk: Node class 'ESS01' is now configured to be recovery group servers.
mmvdisk: Restarting GPFS daemon on node 'ess01io1.gpfs.net'.
mmvdisk: Restarting GPFS daemon on node 'ess01io2.gpfs.net'.
```

The **mmvdisk server configure** command verifies that all node class members have the same general type of server disk topology: either a paired recovery group (Elastic Storage Server) disk topology, or a scale-out recovery group (IBM Storage Scale Erasure Code Edition) disk topology, or a shared recovery group (IBM Elastic Storage System 3000) disk topology. Different IBM Storage Scale configuration settings are applied depending on which type is found.

Important differences between the two configurations are that the paired recovery group pagepool and nsdRAIDTracks settings depend on the amount of real memory, but the shared and scale-out recovery group pagepool defaults to 20% of real memory and nsdRAIDTracks is always 131072 (128K). Regardless of recovery group type, the minimum pagepool setting for a server is 8 GiB.

Table 2. Default paired recovery group pagepool and nsdRAIDTracks settings

Real memory	**pagepool**	**nsdRAIDTracks**
Between 32 GiB and 127 GiB	60% of real memory	131072 (128K)
Between 127 GiB and 256 GiB	70% of real memory	262144 (256K)
Greater than 256 GiB	70% of real memory	327680 (320K)

The default pagepool setting for an mmvdisk node class can be changed with the **--pagepool** option when using the **mmvdisk server configure** command. There is not a similar option for the nsdRAIDTracks setting.

The **mmvdisk** command uses the mmvdisk node class to maintain the IBM Storage Scale RAID configuration settings for the recovery group servers. The **mmvdisk** command never uses cluster-wide settings for recovery group servers, since different recovery groups can use different settings. If recovery group server settings are ever made independently of **mmvdisk**, the same practice of using the mmvdisk node class should be followed.

When a server is added to a scale-out recovery group, the server must first be temporarily configured independently of the recovery group's node class. This is accomplished by using the **-N Node** option to the **mmvdisk server configure** command and restarting the daemon on the node. When the node is added to the scale-out recovery group, it becomes a member of the node class and the temporary

configuration is removed. For more information on adding a server to a scale-out recovery group, see “Recovery group management” on page 29.

Recovery group management

A recovery group is a collection of servers and pdisks.

Scale-out recovery groups have from 4 to 32 identically equipped servers, and each server contributes the same number and type of pdisks to the recovery group.

Shared recovery groups have two servers, and each server simultaneously accesses all of the pdisks in the enclosure twin-tailed between the two servers.

Paired recovery groups have two servers, one primary and one backup, and each recovery group of a pair owns half of the pdisks from the enclosures twin-tailed between the two servers.

In any recovery group, pdisks of the same size and hardware type are collected into declustered arrays. In this context, declustered array means a user declustered array, which is a declustered array where vdisk sets may be defined and where vdisk NSDs are created. Paired recovery groups also have log-only declustered arrays that only contain log tip vdisks and can never contain vdisk sets and vdisk NSDs.

For information on adding disks in the declustered array of the recovery group, see the topic *Adding new disks in the declustered array of the recovery group* in the *IBM Storage Scale Erasure Code Edition* Guide.

Declustered array naming

The user declustered arrays in a recovery group are named DA1, DA2, and so on.

The rules for determining the names given to a declustered array are governed by the size and rotation rate of the pdisks in the declustered array. NVMe and SSD disk devices have a rotation rate of 0, and HDD disk devices typically have rotation rates of 10500 or 7200. The largest size pdisks with the numerically largest rotation rate are placed in DA1. The next largest pdisks by size and rotation rate will go into DA2. If two pdisks are the same size but have different rotation rates, the numerically larger rotation rate will be assigned the first available declustered array number, and the smaller rotation rate will get the declustered array number after that.

Declustered array capacity

The **mmvdisk** command keeps track of the total raw capacity in each user declustered array. The capacity is raw in that it does not account for RAID code redundancy and vdisk metadata overhead. This is the capacity available for defining vdisk sets.

The total raw capacity of a declustered array depends on several factors. The most important factors are the number and size of the pdisks, and the number of equivalent pdisk spares in the declustered array. The size of the pdisks in turn affects the pdisk partition size and number of partitions per pdisk, which also affects the total raw capacity. A portion of each pdisk is reserved for log group atomic updates and for other metadata.

The presence of log home vdisks in a user declustered array also subtracts from the total raw capacity.

The **mmvdisk** command displays the total and free raw capacity of user declustered arrays in several output listings. The most useful is the declustered array section of the **mmvdisk vdiskset list** command, which is used for sizing vdisk sets into declustered arrays:

```
# mmvdisk vdiskset list --recovery-group RG01

                declustered                 capacity            all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
RG01            DA1          SSD    2288 GiB  1144 GiB    50%  DA1.VS1
RG01            DA2          HDD    3330 GiB   833 GiB    25%  DA2.VS1

mmvdisk: Total capacity is the raw space before any vdisk set definitions.
mmvdisk: Free capacity is what remains for additional vdisk set definitions.
```

The **mmvdisk recoverygroup list --declustered-array** command gives a different view that omits the vdisk sets, but includes the number of vdisks, pdisks, and spares:

```
# mmvdisk recoverygroup list --recovery-group RG01 --declustered-array

declustered   needs          vdisks      pdisks     replace        capacity
   array     service  type  user log  total spare  threshold  total raw free raw  background
task
-----------  -------  ----  ---- ---  ----- -----  ---------  --------- --------
---------------
DA1          no       SSD      8   9      8     2          2   2288 GiB 1144 GiB  scrub 14d
(87%)
DA2          no       HDD      8   0     20     3          2   3330 GiB  833 GiB  scrub 14d
(86%)

mmvdisk: Total capacity is the raw space before any vdisk set definitions.
mmvdisk: Free capacity is what remains for additional vdisk set definitions.
```

Declustered array spares

For rebuilding from disk failures, the **mmvdisk** command configures spare space in each declustered array. This space is designated in whole pdisk units, but it is actually spread out across all of the pdisks of the declustered array.

As declustered arrays grow and shrink, the spare space is automatically adjusted to accommodate the new capacity and the existing vdisk sets.

For paired and shared recovery groups using twin-tailed disk enclosures, the number of spares depends on the number of pdisks:

- One spare per enclosure when the total number of pdisks in the declustered array is greater than 24.
- Two spares when the total number of pdisks is between 13 and 24.
- One spare when the total number of pdisks is 12.

For scale-out recovery groups, the rules are more complicated since the declustered arrays can grow and shrink in units dependent on the number of pdisks each server contributes to each declustered array. In the default scale-out declustered array spares management:

- The minimum number of equivalent spares in any scale-out declustered array is two. This permits IBM Storage Scale RAID to always be able to rebuild from two disk failures.
- The maximum number of equivalent spares in any declustered array is two times the number of pdisks an individual server contributes to the declustered array. This permits IBM Storage Scale RAID to eventually be able to rebuild from two server failures.
- As servers are added to scale-out recovery groups, the default number of spares automatically and gradually increases to approach the default maximum.

For scale-out recovery groups, the automatic default spares management can be overridden using the **mmvdisk recoverygroup change** command for all or some of the declustered arrays. Custom spares management can only be used to provide greater rebuild protection than default spares management; a scale-out recovery group cannot be customized to use fewer spares than would be configured by default management. Furthermore, customized spare settings are static and are not automatically adjusted when servers are added to or deleted from a scale-out recovery group. Depending on the vdisk sets defined in the scale-out recovery group, this can require manual adjustment of the custom spares when servers are added or deleted.

Custom spares for the declustered arrays of scale-out recovery groups can be specified in terms of:

- Spare pdisks, meaning the exact number of equivalent pdisk spares to reserve; or
- Spare nodes, meaning the number of pdisks contributed to the declustered array by a specified number of server nodes. This permits the declustered array to rebuild from the specified number of server failures.

The minimum number of custom spares in a scale-out declustered array can never be less than the default number of spares. The maximum number of custom spares in a scale-out declustered array is roughly equal to the number of pdisks supplied by one-third of the servers in the recovery group.

For more information on setting custom spares for scale-out recovery groups, see the **mmvdisk recoverygroup** man page.

Minimum declustered array requirements

A declustered array that was initially created with 9 or more non-spare pdisks is called a large declustered array.

Every recovery group must have at least one large declustered array. A large declustered array will always contain vdisk configuration data replicas and may never be reduced in size below 9 non-spare pdisks. Conversely, a declustered array that was created with fewer than 9 non-spare pdisks will never be used to contain VCD replicas regardless of how large it grows.

This is reflected in the configuration data section of the **mmvdisk recoverygroup list --fault-tolerance** output:

```
# mmvdisk recoverygroup list --recovery-group RG01 --fault-tolerance

                    declustered      VCD spares
configuration data     array     configured  actual  remarks
------------------  -----------  ----------  ------  -------
relocation space    DA2                  11      15  must contain VCD
relocation space    DA1                   0       -  can never contain VCD
```

Log groups

Every recovery group contains one or more log groups. A log group is a collection of vdisks that share the same log home vdisk (the RAID transaction log).

Paired recovery groups have only one log group. A paired recovery group is therefore identical with its single log group. Workload is balanced in a recovery group pair by making each server primary for one of the paired recovery groups and backup for the other.

Scale-out and shared recovery groups have two log groups per server, plus a special root log group that is only used for recovery group metadata and event logs. Workload is balanced among the servers of scale-out and shared recovery groups by making each server responsible for the vdisk NSDs of two log groups.

Vdisk sets balance workload across recovery groups by placing one member vdisk NSD in each log group of each recovery group in which the vdisk set is defined.

The combination of log groups and vdisk sets automatically ensures several IBM Storage Scale RAID best practices:

- Each vdisk set in a file system is balanced across the servers for each recovery group in the vdisk set.
- Each log group of a recovery group contains one member of each vdisk set in the recovery group.
- Each log group of a recovery group therefore represents the same quantity of NSD I/O responsibility.
- Each log group of a recovery group therefore uses the same capacity from each declustered array in the recovery group.
- In the case of scale-out or shared recovery groups, each server is assigned two log groups.
- In the case of a recovery group pair, each server is primary for one log group (one paired recovery group).
- Therefore, each server is nominally responsible for an equal quantity of I/O load and disk capacity across the recovery group or recovery group pair.

The balance of log group assignments can be shown with the **mmvdisk server list** or **mmvdisk recoverygroup list** commands.

For paired recovery groups, the balance is best shown using **mmvdisk server list** with the node class for the recovery group pair since the node class spans both recovery groups of the pair:

```
# mmvdisk server list --node-class ESS01

 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  ess01io1.gpfs.net               yes      serving ESS01L
     2  ess01io2.gpfs.net               yes      serving ESS01R
```

For a scale-out or shared recovery group, the **mmvdisk recoverygroup list** command with the **--server** option can also be used to show all the log groups, since the log groups are contained within a single recovery group:

```
# mmvdisk recoverygroup list --recovery-group RG01 --server

 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  server01                        yes      serving RG01: LG004, LG008
     2  server02                        yes      serving RG01: LG002, LG006
     3  server03                        yes      serving RG01: LG003, LG007
     4  server04                        yes      serving RG01: root, LG001, LG005
```

Recovery group server maintenance

It is sometimes necessary to temporarily take an IBM Storage Scale RAID recovery group server off-line for software or hardware maintenance, while at the same time preserving recovery group access.

For the two servers of paired recovery groups, this is accomplished using the **mmvdisk recoverygroup change --active** command to make one server the active server for both recovery groups of the pair. Maintenance can then be performed on the non-active server. Since the active server has access to all file system data in the two paired recovery groups, the paired recovery group building block can run indefinitely on the active server, albeit with lower performance and no standby server should the active server fail.

When maintenance is finished on a paired recovery group server and it is restarted, the maintenance procedure can be repeated for the other server by making the restarted server the active server for both recovery groups.

For the 4 to 32 servers of scale-out recovery groups, the pdisks and file system data on each server are only available when the server is up. When a server is off-line, IBM Storage Scale RAID must use RAID fault tolerance to reconstruct any file system data stored on that server. If a server stays off-line for more than a default of 20 minutes, IBM Storage Scale RAID will begin rebuilding the server's data onto spare space. Furthermore, if multiple servers are off-line at the same time, fault tolerance can be exceeded and file system data can become temporarily unavailable until the servers are brought back on-line. This makes maintenance considerations very different for scale-out servers.

Maintenance for scale-out recovery group servers must be performed using the **mmvdisk recoverygroup change --suspend** and **mmvdisk recoverygroup change --resume** commands. The suspend command safely stops IBM Storage Scale RAID on a scale-out server by checking that no other servers are suspended and that enough fault tolerance exists in the recovery group to permit taking the server off-line. A server is suspended by suspending all of its pdisks and then making it ineligible to serve log groups. The resume command brings a scale-out server back on-line by enabling it to serve log groups and resuming all of its pdisks.

Scale-out server software or hardware maintenance can safely be accomplished by sequentially:

- Suspending one scale-out server
- Performing the desired maintenance on that server
- Resuming that server when maintenance is completed
- Repeating the procedure for the next server

Note: Scale-out server maintenance should be performed quickly to avoid unnecessary RAID rebuild overhead. By default, there is a window of 20 minutes before the data on a suspended server's pdisks is rebuilt onto other nodes. The **mmvdisk recoverygroup change --suspend** command has a **--window N** option to adjust the number of minutes before the suspended server's pdisks are rebuilt. The value of **N** must be a number between 10 and 60 inclusive.

The **mmvdisk** command considers a scale-out server to be suspended if it is not eligible to serve log groups or if all of its pdisks are suspended. The **mmvdisk recoverygroup list --server** command will show any suspended servers. The **mmvdisk recoverygroup change --suspend** command will not allow more than one server to be suspended. The **mmvdisk recoverygroup change --resume** command can be used to resume any scale-out server, even one that is not suspended; this has the effect of starting IBM Storage Scale RAID on the server, making sure it is eligible to serve log groups, and resuming any of its pdisks that are suspended.

For more information on suspending and resuming scale-out recovery group servers, see the **mmvdisk recoverygroup** man page.

Vdisk set management

A vdisk set is a collection of uniform vdisk NSDs from one or more mmvdisk recovery groups. Each log group of a recovery group included in a vdisk set contributes one identical member vdisk NSD.

Elements of a vdisk set definition

The definition of a vdisk set requires seven attributes and one or more recovery groups:

1. **Vdisk set name**: A name for the vdisk set that is unique within the IBM Storage Scale cluster.
2. **Declustered array**: The declustered array in the recovery groups where the member vdisk NSDs are located.
3. **RAID code**: The vdisk RAID code used for the member vdisk NSDs.
4. **Block size**: The file system block size (equivalently, the vdisk track size) of the member vdisk NSDs.
5. **Vdisk NSD size**: The usable size of a member vdisk NSD.
6. **NSD usage**: The file system usage of a member vdisk NSD.
7. **Storage pool**: The file system storage pool of a member vdisk NSD.

   Recovery groups: One or more recovery groups, each of which contributes one member vdisk NSD to the vdisk set.

Defining a vdisk set requires specifying at least one recovery group, the vdisk set name, the RAID code, the block size, and the vdisk set size (from which the member vdisk NSD size is calculated). The declustered array, NSD usage, and the storage pool attributes can be omitted when **mmvdisk** can determine suitable default values.

Table 3. Vdisk set attributes and their default values

Vdisk set attribute	Default value	Valid values
vdisk set name	must be specified	The vdisk set name must be unique within the IBM Storage Scale cluster. The name is limited to the same character set as IBM Storage Scale file system device names and IBM Storage Scale RAID recovery group names.

Table 3. Vdisk set attributes and their default values (continued)

Vdisk set attribute	Default value	Valid values
declustered array name	DA1 (if only one DA is present)	All recovery groups for which this vdisk set is defined must have a declustered array with this name where this vdisk set's members are created. The expectation is that a vdisk set extends across structurally identical recovery groups where the named declustered array has the same characteristics in each recovery group. If there is only one user declustered array in each recovery group, it is named DA1 and this is the default. If there is more than one user declustered array in a recovery group, there is no default and a declustered array name must be specified.
RAID code	must be specified	This is the vdisk RAID code for members of the vdisk set. Valid values are: 3WayReplication, 4WayReplication, 4+2P, 4+3P, 8+2P, or 8+3P
block size	must be specified	This is the file system block size (vdisk track size) for members of the vdisk set. It is constrained by the selected RAID code. Valid values for 3WayReplication and 4WayReplication are 256k, 512k, 1m, or 2m. Valid values for 4+2P and 4+3P are 512k, 1m, 2m, 4m, or 8m. Valid values for 8+2P and 8+3P are 512k, 1m, 2m, 4m, 8m, or 16m.
vdisk set size	must be specified	The vdisk set size is the desired aggregate size of the vdisk set members in one recovery group. The set size can be specified as a percentage (whole numbers from 1% to 100% using the % suffix) or as a number of bytes (a number, optionally followed by one of the base 2 suffixes K, M, G, or T). If the vdisk set size is given as a percentage, it specifies the raw size to use from the declustered array including RAID code redundancy. If the vdisk set size is given as a number of bytes, it specifies the desired usable size of the vdisk set excluding RAID code redundancy. The vdisk set size is used to calculate the usable size of a single vdisk NSD member of the vdisk set in one recovery group. It is this calculated usable size that becomes part of the vdisk set definition, so if the size of a declustered array should ever change, the size of the individual member vdisk NSDs remains constant. **Note:** The resulting actual set size will almost certainly differ from the requested set size, especially if a number of usable bytes is specified, since the IBM Storage Scale RAID internal data structures to accommodate the requested size will depend on the number and size of the pdisks and on the RAID code and block size of the member vdisks.
NSD usage	dataAndMetadata	This is the IBM Storage Scale file system data usage for the NSD. Valid values are dataAndMetadata, metadataOnly, and dataOnly. The default is dataAndMetadata.
storage pool	system	If the NSD usage is dataAndMetadata or metadataOnly, the storage pool value must be `system` and does not need to be specified. If the NSD usage is dataOnly, the storage pool must be specified and the value may not be `system`.

The following illustration is a visualization of how four vdisk sets might span across four paired recovery groups (two ESS recovery group pairs), and how they are used to form three file systems:

```
                                File System           File System   File System
                                    FSA                   FSB           FSC
                              /-----------------------\   /---------\   /---------\
                               Vdisk Set     Vdisk Set     Vdisk Set     Vdisk Set
                                 VSA1          VSA2          VSB           VSC
                              /--------\    /--------\    /-------\     /-------\
  Recovery Group ESS01L {     VSA1 vdisk  |  VSA2 vdisk  |  VSB vdisk  |  VSC vdisk  }
  Recovery Group ESS01R {     VSA1 vdisk  |  VSA2 vdisk  |  VSB vdisk  |  VSC vdisk  }
  Recovery Group ESS02L {     VSA1 vdisk  |  VSA2 vdisk  |  VSB vdisk  |  VSC vdisk  }
  Recovery Group ESS02R {     VSA1 vdisk  |  VSA2 vdisk  |  VSB vdisk  |  VSC vdisk  }
                              \--------/    \--------/    \-------/     \-------/
                               Vdisk Set     Vdisk Set     Vdisk Set     Vdisk Set
                                 VSA1          VSA2          VSB           VSC
                              \-----------------------/   \---------/   \---------/
                                    File System           File System   File System
                                        FSA                   FSB           FSC
```

There are four structurally identical recovery groups built from two ESS building blocks.

There are three file systems all uniformly striped across the recovery groups using four vdisk sets. An identical member of each of the four sets vdisk sets is present in each recovery group.

File system FSA uses two vdisk sets, VSA1 and VSA2.

File system FSB uses one vdisk set, VSB.

File system FSC uses one vdisk set, VSC.

For the high-level purpose of visualizing how four vdisk sets might span four recovery groups to construct three file systems, details such as the RAID code and block size for the four vdisk sets are omitted. It is enough to know that within a vdisk set, the details are the same for each member vdisk NSD.

This illustration also shows how the concept of vdisk sets helps to accomplish the IBM Storage Scale RAID best practice of constructing file systems from identical vdisk NSDs equally distributed across all recovery groups.

Vdisk set definition sizing and preview

Defining vdisk sets before actually creating the member vdisk NSDs permits the IBM Storage Scale administrator to preview the declustered array size and server memory requirements of the vdisk sets.

The **mmvdisk** command refuses to define a vdisk set that exceeds declustered array disk capacity in a recovery group or server memory capacity in the recovery group server node class.

Newly created recovery groups have no vdisk sets. To see what capacity the newly created recovery groups have for defining vdisk sets, use **mmvdisk vdiskset list --recovery-group all**. This command shows all recovery group declustered array capacities where vdisk sets can be defined, and the server memory capacities for the recovery group server node classes.

In this example, the two recovery groups, which are newly created, have 100% of their raw declustered array capacity free. Their servers are using a baseline 386 MiB of 30 GiB available memory for internal server vdisk set maps.

```
# mmvdisk recoverygroup list

                                                          needs    user
recovery group  active   current or master server        service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
BB01L           yes      server01.gpfs.net                no            0
BB01R           yes      server02.gpfs.net                no            0

# mmvdisk vdiskset list --recovery-group all

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
BB01L           DA1             36 TiB    36 TiB   100%  -
```

```
BB01R           DA1             36 TiB    36 TiB   100%  -

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01           30 GiB   386 MiB  -

#
```

As vdisk sets are defined in these recovery groups, **mmvdisk** shows the cumulative claim made by the vdisk sets upon the declustered array capacity and server memory resources. It does so without actually having to create the vdisk NSDs.

The next example shows the accumulating declustered array space and memory claims associated with the definition of three vdisk sets in the BB01L and BB01R recovery groups and the BB01 server node class.

```
# mmvdisk vdiskset define --vdisk-set fs1data --recovery-group BB01L,BB01R --set-size 45% \
                          --code 8+3p --block-size 2m --nsd-usage dataOnly --storage-pool data
mmvdisk: Vdisk set 'fs1data' has been defined.
mmvdisk: Recovery group 'BB01L' has been defined in vdisk set 'fs1data'.
mmvdisk: Recovery group 'BB01R' has been defined in vdisk set 'fs1data'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
fs1data             2   11 TiB   16 TiB  no       -, DA1, 8+3p, 2 MiB, dataOnly, data

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
BB01L           DA1             36 TiB    20 TiB    55%  fs1data
BB01R           DA1             36 TiB    20 TiB    55%  fs1data

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01           30 GiB   869 MiB  fs1data (483 MiB)

# mmvdisk vdiskset define --vdisk-set fs1meta --recovery-group BB01L,BB01R --set-size 5% \
                          --code 4wayReplication --block-size 512k --nsd-usage metadataOnly
mmvdisk: Vdisk set 'fs1meta' has been defined.
mmvdisk: Recovery group 'BB01L' has been defined in vdisk set 'fs1meta'.
mmvdisk: Recovery group 'BB01R' has been defined in vdisk set 'fs1meta'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
fs1meta             2  463 GiB 1872 GiB  no       -, DA1, 4WayReplication, 512 KiB,
metadataOnly, system

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
BB01L           DA1             36 TiB    18 TiB    50%  fs1data, fs1meta
BB01R           DA1             36 TiB    18 TiB    50%  fs1data, fs1meta

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01           30 GiB   908 MiB  fs1data (483 MiB), fs1meta (38 MiB)

# mmvdisk vdiskset define --vdisk-set fs2 --recovery-group BB01L,BB01R --set-size 50% \
                          --code 8+3p --block-size 2m
mmvdisk: Vdisk set 'fs2' has been defined.
mmvdisk: Recovery group 'BB01L' has been defined in vdisk set 'fs2'.
mmvdisk: Recovery group 'BB01R' has been defined in vdisk set 'fs2'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
fs2                 2   13 TiB   18 TiB  no       -, DA1, 8+3p, 2 MiB, dataAndMetadata, system

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
BB01L           DA1             36 TiB  4096 MiB     0%  fs1data, fs1meta, fs2
BB01R           DA1             36 TiB  4096 MiB     0%  fs1data, fs1meta, fs2
```

```
                 vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01           30 GiB  1445 MiB  fs1data (483 MiB), fs1meta (38 MiB), fs2 (537 MiB)
```

At this point, 100% of the disk capacity in DA1 has been claimed by these vdisk set definitions. The server memory demand is well under the available limit. None of the vdisk sets have been created yet. A vdisk set is only created when all of the expected vdisk NSDs in all the vdisk set's recovery groups exist in reality and not merely as a claim upon resource. If none, or only some but not all, of the expected vdisk NSDs actually exist, the value in the `created` column for a vdisk set is no.

Assuming these definitions are what is desired, the next example shows actually creating these three vdisk sets.

```
# mmvdisk vdiskset create --vdisk-set all
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'fs2'.
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'fs1data'.
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'fs1meta'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS003
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS003
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS002
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS002
mmvdisk: Created all vdisks in vdisk set 'fs2'.
mmvdisk: Created all vdisks in vdisk set 'fs1data'.
mmvdisk: Created all vdisks in vdisk set 'fs1meta'.
mmvdisk: (mmcrnsd) Processing disk RG001VS003
mmvdisk: (mmcrnsd) Processing disk RG002VS003
mmvdisk: (mmcrnsd) Processing disk RG001VS001
mmvdisk: (mmcrnsd) Processing disk RG002VS001
mmvdisk: (mmcrnsd) Processing disk RG001VS002
mmvdisk: (mmcrnsd) Processing disk RG002VS002
mmvdisk: Created all NSDs in vdisk set 'fs2'.
mmvdisk: Created all NSDs in vdisk set 'fs1data'.
mmvdisk: Created all NSDs in vdisk set 'fs1meta'.
# mmvdisk vdiskset list --vdisk-set all

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
fs1data             2   11 TiB   16 TiB  yes      -, DA1, 8+3p, 2 MiB, dataOnly, data
fs1meta             2  463 GiB 1872 GiB  yes      -, DA1, 4WayReplication, 512 KiB,
metadataOnly, system
fs2                 2   13 TiB   18 TiB  yes      -, DA1, 8+3p, 2 MiB, dataAndMetadata, system

                declustered           capacity            all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
BB01L           DA1             36 TiB  4096 MiB     0%  fs1data, fs1meta, fs2
BB01R           DA1             36 TiB  4096 MiB     0%  fs1data, fs1meta, fs2

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01           30 GiB  1445 MiB  fs1data (483 MiB), fs1meta (38 MiB), fs2 (537 MiB)

#
```

The **mmvdisk vdiskset create** command shows the creation of the individual member vdisk NSDs in which the automatically generated names of the individual member vdisks and NSDs are seen. The names of individual vdisk NSDs are generally unimportant since normal **mmvdisk** usage manages them collectively as vdisk set members.

Note: If it is ever required to know the details of individual member vdisk NSDs, the **mmvdisk vdisk list** and **mmvdisk recoverygroup list --vdisk** commands can be used.

After vdisk creation, when the vdisk sets are listed again, the `created` column shows `yes` for all three vdisk sets. The actual member vdisk NSDs have been created and are ready to be used to create file systems.

```
# mmvdisk filesystem create --file-system fs1 --vdisk-set fs1meta,fs1data
# mmvdisk filesystem create --file-system fs2 --vdisk-set fs2
```

After file system creation, listing the vdisk sets shows the file system device name with the vdisk set attributes.

```
# mmvdisk vdiskset list --vdisk-set all

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
fs1data             2   11 TiB   16 TiB  yes      fs1, DA1, 8+3p, 2 MiB, dataOnly, data
fs1meta             2  463 GiB 1872 GiB  yes      fs1, DA1, 4WayReplication, 512 KiB,
metadataOnly, system
fs2                 2   13 TiB   18 TiB  yes      fs2, DA1, 8+3p, 2 MiB, dataAndMetadata, system
...
```

File system management

Once vdisk sets are defined and created, the **mmvdisk filesystem create** command will take the vdisk sets and run the core IBM Storage Scale **mmcrfs** command with an appropriate vdisk NSD stanza file and appropriate options based on the vdisk set attributes.

Custom **mmcrfs** options unrelated to the vdisk set attributes may be supplied at file system creation using the **mmvdisk filesystem create --mmcrfs** option. All command line parameters after **--mmcrfs** are passed to the IBM Storage Scale **mmcrfs** command.

- The **mmcrfs** option **-j scatter** is always used since **-j cluster** offers no benefit with vdisk NSDs. If **-j cluster** is used with the **--mmcrfs** option, it is considered an error.
- The **mmcrfs** default replication settings for both data (**-r**) and metadata (**-m**) are set to 1.
- The **mmcrfs** maximum replication settings for both data (**-R**) and metadata (**-M**) are set to 2.

If custom default or maximum replication settings are used with the **--mmcrfs** option, **mmvdisk** will notice them and make sure that the file system failure groups for the vdisk NSDs can be set appropriately. For example, if a default replication of 3 is used in a file system where the vdisk set has only two members, it will be considered an error.

The **mmvdisk filesystem** command uses failure group numbers 1 and 2 when the maximum replication is 2, and failure group numbers 1, 2, and 3 when the maximum replication is 3.

Node classes represent the single point of failure for vdisk set file system NSD failure groups. If the vdisk sets of the file system span sufficient node classes to match or exceed the maximum replication, all the vdisk NSDs in a node class will be assigned the same failure group.

If the vdisk sets do not span sufficient node classes to match the maximum replication, failure groups will be assigned based on log groups within a scale-out or shared recovery group, or based on the individual recovery groups in a recovery group pair. While this does not offer true single point of failure protection, it does allow the file system to replicate, and if recovery groups in sufficient node classes are added to the file system later, the **mmvdisk filesystem change** command can be used to reassign the failure groups based properly on the node classes.

If a file system has sufficient node classes to match or exceed the maximum replication, custom failure group numbers can be assigned to the file system's vdisk NSDs on a node class basis. This can be useful if multiple recovery groups share a single point of failure that **mmvdisk** is unaware of, such as being in the same frame or on the same network switch.

For example, if node classes NC01 and NC02 are in one frame, and node classes NC03 and NC04 are in another, the vdisk NSDs in NC01 and NC02 can be assigned the failure group number 12 and the vdisk NSDs in NC03 and NC04 can be assigned the failure group number 34:

```
# mmvdisk filesystem create --file-system fs1 \
                            --vdisk-set TwoFramesFourNCs \
                            --failure-groups NC01=12,NC02=12,NC03=34,NC04=34
```

In this example, the maximum replication is 2 for both data and metadata, and two custom failure group numbers are used.

For more information and restrictions on custom failure group assignments, see the "mmvdisk filesystem command" on page 373.

Use cases for mmvdisk

The following use cases provide examples of how **mmvdisk** can be used.

A complete use case for mmvdisk

This example assumes that there is a properly installed three-node IBM Storage Scale cluster with an ESS GL6 building block on the server pair ess01io1 and ess01io2:

```
# mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      ess01io1         active
       2      ess01io2         active
       3      ems              active
```

Define an **mmvdisk** node class for the server pair:

```
# mmvdisk nodeclass create --node-class ESS01 -N ess01io1,ess01io2
mmvdisk: Node class 'ESS01' created.
```

Verify that the correct ESS GL6 supported disk topology is detected on the node class:

```
# mmvdisk server list --node-class ESS01 --disk-topology

 node                                    needs    matching
number  server                         attention   metric   disk topology
------  -------------------------------  ---------  --------  -------------
     1  ess01io1.gpfs.net                no          100/100  ESS GL6
     2  ess01io2.gpfs.net                no          100/100  ESS GL6
```

Since a correct disk topology has been detected, configure the node class to be recovery group servers:

```
# mmvdisk server configure --node-class ESS01 --recycle one
mmvdisk: Checking resources for specified nodes.
mmvdisk: Node 'ess01io1.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node 'ess01io2.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node class 'ESS01' has a paired recovery group disk topology.
mmvdisk: Setting configuration for node class 'ESS01'.
mmvdisk: Node class 'ESS01' is now configured to be recovery group servers.
mmvdisk: Restarting GPFS daemon on node 'ess01io1.gpfs.net'.
mmvdisk: Restarting GPFS daemon on node 'ess01io2.gpfs.net'.
```

With the node class configured, create a recovery group pair:

```
# mmvdisk recoverygroup create --node-class ESS01 --recovery-group ESS01L,ESS01R
mmvdisk: Checking node class configuration.
mmvdisk: Checking daemon status on node 'ess01io1.gpfs.net'.
mmvdisk: Checking daemon status on node 'ess01io2.gpfs.net'.
mmvdisk: Analyzing disk topology for node 'ess01io1.gpfs.net'.
mmvdisk: Analyzing disk topology for node 'ess01io2.gpfs.net'.
mmvdisk: Creating recovery group 'ESS01L'.
mmvdisk: Creating recovery group 'ESS01R'.
mmvdisk: Creating log vdisks for recovery groups.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIPBACKUP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIP
```

```
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIPBACKUP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGHOME
mmvdisk: Created recovery groups 'ESS01L' and 'ESS01R'.
```

With the recovery group pair created, the sizing available for vdisk sets can be listed:

```
# mmvdisk vdiskset list --recovery-group all

                declustered           capacity            all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
ESS01L          DA1            911 TiB   911 TiB   100%  -
ESS01R          DA1            911 TiB   911 TiB   100%  -

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB   387 MiB  -
```

For this example, assume that it is desired to use 50% of the space in declustered array DA1 to create a file system using RAID code 8+3p and a 16M block size.

Define vdisk set VS1 in recovery groups ESS01L and ESS01R using RAID code 8+3p, block size 16m, and a vdisk set size of 50%:

```
# mmvdisk vdiskset define --vdisk-set VS1 --recovery-group ESS01L,ESS01R --code 8+3p --block-size 16m --set-size 50%
mmvdisk: Vdisk set 'VS1' has been defined.
mmvdisk: Recovery group 'ESS01L' has been defined in vdisk set 'VS1'.
mmvdisk: Recovery group 'ESS01R' has been defined in vdisk set 'VS1'.

                    member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  --------------------------
VS1                 2  330 TiB  455 TiB  no       -, DA1, 8+3p, 16 MiB, dataAndMetadata, system

                declustered           capacity            all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
ESS01L          DA1            911 TiB   455 TiB    50%  VS1
ESS01R          DA1            911 TiB   455 TiB    50%  VS1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB  8443 MiB  VS1 (8055 MiB)
```

Assuming that the sizes that result from this definition are satisfactory, create the actual member vdisk NSDs for vdisk set VS1:

```
# mmvdisk vdiskset create --vdisk-set VS1
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'VS1'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS001
mmvdisk: Created all vdisks in vdisk set 'VS1'.
mmvdisk: (mmcrnsd) Processing disk RG001VS001
mmvdisk: (mmcrnsd) Processing disk RG002VS001
mmvdisk: Created all NSDs in vdisk set 'VS1'.
```

Then use **mmvdisk filesystem create** to create file system FS1 using vdisk set VS1. In this example, the **--mmcrfs** option is used to pass all remaining command line parameters to the underlying **mmcrfs** command to enable quotas and select a custom file system mount point:

```
# mmvdisk filesystem create --file-system FS1 --vdisk-set VS1 --mmcrfs -Q yes -T /FS1
mmvdisk: Creating file system 'FS1'.
mmvdisk: The following disks of FS1 will be formatted on node ess01io1:
mmvdisk:     RG001VS001: size 346179584 MB
mmvdisk:     RG002VS001: size 346179584 MB
mmvdisk: Formatting file system ...
mmvdisk: Disks up to size 2.58 PB can be added to storage pool system.
mmvdisk: Creating Inode File
mmvdisk:   68 % complete on Tue Jun 12 11:29:10 2018
mmvdisk:  100 % complete on Tue Jun 12 11:29:12 2018
mmvdisk: Creating Allocation Maps
mmvdisk: Creating Log Files
mmvdisk: Clearing Inode Allocation Map
mmvdisk: Clearing Block Allocation Map
mmvdisk: Formatting Allocation Map for storage pool system
mmvdisk: Completed creation of file system /dev/FS1.
```

The file system is now ready to be mounted and used:

```
# mmvdisk filesystem list --file-system FS1

                                vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS1             ESS01L              1               1  yes       yes    system
VS1             ESS01R              1               2  yes       yes    system

# mmmount FS1 -a
Tue Jun 12 11:37:28 EDT 2018: mmmount: Mounting file systems ...
# sync; df /FS1
Filesystem        1K-blocks    Used    Available Use% Mounted on
FS1            708975788032 5718016 708970070016   1% /FS1
```

Use case for multiple identical building blocks

This example demonstrates the best practice for using vdisk sets with multiple identical ESS building blocks.

This example assumes that there are two identical GL1S ESS building blocks. The four recovery groups (two per building block) will have an identical structure with a single user declustered array (DA1), which has the same number and size of disks. Defining a vdisk set in DA1 across the four recovery groups guarantees that each recovery group has a vdisk with the same attributes. The vdisk set can then be used to create a file system that is striped uniformly across the recovery groups.

In this example, **mmvdisk** abbreviations will be used to demonstrate their convenience.

The building blocks and the recovery groups can be identified by listing the **mmvdisk** node classes:

```
# mmvdisk nc list

node class  recovery groups
----------  ---------------
BB01        BB01L, BB01R
BB02        BB02L, BB02R
```

The identical ESS building block topologies can be seen by listing the server disk topology for each node class:

```
# mmvdisk server list --nc BB01 --disk-topology

 node                                    needs    matching
number  server                          attention   metric   disk topology
------  ------------------------------  ---------  --------  -------------
     1  server01.gpfs.net               no          100/100  ESS GL1S
     2  server02.gpfs.net               no          100/100  ESS GL1S

# mmvdisk server list --nc BB02 --disk-topology

 node                                    needs    matching
number  server                          attention   metric   disk topology
------  ------------------------------  ---------  --------  -------------
     3  server03.gpfs.net               no          100/100  ESS GL1S
     4  server04.gpfs.net               no          100/100  ESS GL1S
```

Sizing information for vdisk sets in the recovery groups is shown with the **mmvdisk vdiskset list** command. When given the **--recovery-group all** parameter, it shows all of the user declustered arrays and their capacities as well as all of the server node classes and their memory requirements:

```
# mmvdisk vs list --rg all

               declustered                capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
BB01L           DA1          HDD      18 TiB    18 TiB   100%  -
BB01R           DA1          HDD      18 TiB    18 TiB   100%  -
BB02L           DA1          HDD      18 TiB    18 TiB   100%  -
BB02R           DA1          HDD      18 TiB    18 TiB   100%  -

                 vdisk set map memory per server
node class  available  required  required per vdisk set
```

```
----------  ---------  --------  ----------------------
BB01         4702 MiB   385 MiB  -
BB02         4702 MiB   385 MiB  -
```

In this example, assume that you want to add a file system (fs1) that uses 50% of the raw DA1 declustered array capacity. A vdisk set for this file system can be defined to claim 50% of DA1 in each recovery group. To aid in identification, the vdisk set can be given the same name as the desired file system. Since all of the recovery groups are identical, the **all** keyword can be used to select the recovery groups for the vdisk set. The other elements of a vdisk set definition are:

- the declustered array (**--declustered-array**, **--da**) in each recovery group
- the size of the vdisk set (**--set-size**, **--ss**) in each declustered array
- the RAID code (**--code**) for each vdisk NSD
- the block size (**--block-size**, **--bs**) for each vdisk NSD
- the file system NSD usage (**--nsd-usage**, **--usage**) for each vdisk NSD
- the file system storage pool (**--storage-pool**, **--pool**) for each vdisk NSD

In this example, since each recovery group has a single DA1 declustered array, **mmvdisk** will use it as the default. The file system NSD usage defaults to **dataAndMetadata** and the storage pool defaults to **system**. This leaves the choice of RAID code, block size, and vdisk set size to be specified:

```
# mmvdisk vs define --vs fs1 --rg all --ss 50% --code 8+3p --bs 16m
mmvdisk: Vdisk set 'fs1' has been defined.
mmvdisk: Recovery group 'BB01L' has been defined in vdisk set 'fs1'.
mmvdisk: Recovery group 'BB01R' has been defined in vdisk set 'fs1'.
mmvdisk: Recovery group 'BB02L' has been defined in vdisk set 'fs1'.
mmvdisk: Recovery group 'BB02R' has been defined in vdisk set 'fs1'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
fs1                 4 6993 GiB 9702 GiB  no       -, DA1, 8+3p, 16 MiB, dataAndMetadata, system

                declustered                 capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
BB01L           DA1          HDD      18 TiB  9707 GiB    50%  fs1
BB01R           DA1          HDD      18 TiB  9707 GiB    50%  fs1
BB02L           DA1          HDD      18 TiB  9707 GiB    50%  fs1
BB02R           DA1          HDD      18 TiB  9707 GiB    50%  fs1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB01         4702 MiB   552 MiB  fs1 (167 MiB)
BB02         4702 MiB   552 MiB  fs1 (167 MiB)
```

After validating the parameters of the vdisk set definition, the **mmvdisk vdiskset define** command will display the attributes of the new definition and the claim that the definition makes upon declustered array capacity and server memory. This information permits the storage administrator to evaluate the attributes and the claim on resources.

To create the member vdisk NSDs of the vdisk set according to the definition, the **mmvdisk vdiskset create** command is used:

```
# mmvdisk vs create --vs fs1
mmvdisk: 4 vdisks and 4 NSDs will be created in vdisk set 'fs1'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG003VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG004VS001
mmvdisk: Created all vdisks in vdisk set 'fs1'.
mmvdisk: (mmcrnsd) Processing disk RG001VS001
mmvdisk: (mmcrnsd) Processing disk RG002VS001
mmvdisk: (mmcrnsd) Processing disk RG003VS001
mmvdisk: (mmcrnsd) Processing disk RG004VS001
mmvdisk: Created all NSDs in vdisk set 'fs1'.
```

Once the member vdisk NSDs of the vdisk set are created, the file system is created using the vdisk set. The **mmvdisk filesystem create** command uses the block size, storage pool, and NSD usage information from the vdisk set and supplies them to the IBM Storage Scale **mmcrfs** command, together with the **-j scatter** best practice for IBM Storage Scale RAID file systems. File system NSD failure groups are assigned per building block when multiple building blocks are used. Other parameters can be passed to the **mmcrfs** command using the **--mmcrfs** parameter:

```
# mmvdisk fs create --fs fs1 --vs fs1 --mmcrfs -A no -Q yes -T /fs1
mmvdisk: Creating file system 'fs1'.
mmvdisk: The following disks of fs1 will be formatted on node kvm5:
mmvdisk:     RG001VS001: size 7161712 MB
mmvdisk:     RG002VS001: size 7161712 MB
mmvdisk:     RG003VS001: size 7161712 MB
mmvdisk:     RG004VS001: size 7161712 MB
mmvdisk: Formatting file system ...
mmvdisk: Disks up to size 66.81 TB can be added to storage pool system.
mmvdisk: Creating Inode File
mmvdisk:   34 % complete on Tue Nov  6 19:59:49 2018
mmvdisk:   74 % complete on Tue Nov  6 19:59:54 2018
mmvdisk:  100 % complete on Tue Nov  6 19:59:57 2018
mmvdisk: Creating Allocation Maps
mmvdisk: Creating Log Files
mmvdisk:   75 % complete on Tue Nov  6 20:00:02 2018
mmvdisk:  100 % complete on Tue Nov  6 20:00:03 2018
mmvdisk: Clearing Inode Allocation Map
mmvdisk: Clearing Block Allocation Map
mmvdisk: Formatting Allocation Map for storage pool system
mmvdisk:   74 % complete on Tue Nov  6 20:00:09 2018
mmvdisk:  100 % complete on Tue Nov  6 20:00:12 2018
mmvdisk: Completed creation of file system /dev/fs1.
```

The **mmvdisk filesystem list** command can be used to show that the newly created file system is striped uniformly across the recovery groups:

```
# mmvdisk fs list --fs fs1

                                  vdisk     list of       holds    holds  storage
vdisk set       recovery group    count  failure groups  metadata  data    pool
--------------  --------------    -----  --------------  --------  -----  -------
fs1             BB01L                1               1  yes       yes    system
fs1             BB01R                1               1  yes       yes    system
fs1             BB02L                1               2  yes       yes    system
fs1             BB02R                1               2  yes       yes    system
```

Reporting file system orphans with mmvdisk

A vdisk set must belong to a single file system in its entirety.

If a file system vdisk set is extended into additional recovery groups, a situation is created where some of the member vdisk NSDs are in the file system and some are not. The vdisk NSDs that are not yet in the file system to which they belong are considered "orphaned" from the file system.

A similar but less common situation arises when a recovery group is in the process of being removed from a vdisk set. The member vdisk NSDs from that recovery group are first deleted from a file system, but they remain in the vdisk set until they are deleted from the recovery group and the recovery group is removed from the vdisk set definition.

The **mmvdisk** command detects these situations and reminds the administrator that more work is required to bring the vdisk set and the file system into alignment.

The following example illustrates how **mmvdisk** reports file system orphans after a vdisk set has been defined and created in one recovery group and used to create a file system:

```
# mmvdisk vdiskset list --vdisk-set VS1

                    member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
VS1                 1  132 TiB  182 TiB  yes      FS1, DA1, 8+3p, 16 MiB, dataAndMetadata,
system

                declustered           capacity            all vdisk sets defined
```

```
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
ESS01L          DA1            911 TiB   729 TiB    80%  VS1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB  1998 MiB  VS1 (1610 MiB)

# mmvdisk filesystem list --file-system FS1

                                 vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS1             ESS01L              1               1  yes       yes    system
```

It is now desired to extend the vdisk set VS1 and the file system FS1 into recovery group ESS01R.

When recovery group ESS01R is added to the VS1 vdisk set definition, **mmvdisk** immediately notices that the file system FS1 is no longer in sync with vdisk set VS1:

```
# mmvdisk vdiskset define --vdisk-set VS1 --recovery-group ESS01R
mmvdisk: Recovery group 'ESS01R' has been defined in vdisk set 'VS1'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  --------------------------
VS1                 2  132 TiB  182 TiB  no       FS1, DA1, 8+3p, 16 MiB, dataAndMetadata,
system

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
ESS01L          DA1            911 TiB   729 TiB    80%  VS1
ESS01R          DA1            911 TiB   729 TiB    80%  VS1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB  3609 MiB  VS1 (3221 MiB)

mmvdisk: Attention: There are incomplete vdisk set or file system changes.
mmvdisk: Members of vdisk set 'VS1' are orphaned from file system 'FS1'.
mmvdisk: Complete any vdisk set or file system changes to dismiss this notice.
```

When a file system or vdisk set in this state is listed, **mmvdisk** reminds the administrator that work remains to be done. Listing file system FS1 provides another illustration:

```
# mmvdisk filesystem list --file-system FS1

                                 vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS1             ESS01L              1               1  yes       yes    system
VS1             ESS01R              0               -  yes       yes    system

mmvdisk: Attention: There are incomplete vdisk set or file system changes.
mmvdisk: Members of vdisk set 'VS1' are orphaned from file system 'FS1'.
mmvdisk: Complete any vdisk set or file system changes to dismiss this notice.
```

In this case, since the intention is to extend file system FS1 into recovery group ESS01R, completing the vdisk set and file system changes means creating the new member vdisk NSDs for VS1 and adding them to the file system:

```
# mmvdisk vdiskset create --vdisk-set VS1
mmvdisk: 1 vdisk and 1 NSD will be created in vdisk set 'VS1'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS001
mmvdisk: Created all vdisks in vdisk set 'VS1'.
mmvdisk: (mmcrnsd) Processing disk RG002VS001
mmvdisk: Created all NSDs in vdisk set 'VS1'.

mmvdisk: Attention: There are incomplete vdisk set or file system changes.
mmvdisk: Members of vdisk set 'VS1' are orphaned from file system 'FS1'.
mmvdisk: Complete any vdisk set or file system changes to dismiss this notice.
```

The new member vdisk NSDs have been created, but they are still orphaned because they must also be added to the file system:

```
# mmvdisk filesystem add --file-system FS1 --vdisk-set VS1
mmvdisk: The following disks of FS1 will be formatted on node ess01io1:
mmvdisk:     RG002VS001: size 138425984 MB
mmvdisk: Extending Allocation Map
mmvdisk: Checking Allocation Map for storage pool system
mmvdisk: Completed adding disks to file system FS1.
```

Now that the new member vdisk NSDs have been added to the file system, listing the file system no longer shows the notice about orphaned members:

```
# mmvdisk filesystem list --file-system FS1

                                vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS1             ESS01L              1               1  yes       yes    system
VS1             ESS01R              1               2  yes       yes    system
```

Note: This output showed that **mmvdisk** automatically followed file system best practices in using a different file system NSD failure group number for the new vdisk NSD from recovery group ESS01R.

Converting existing recovery groups to mmvdisk management

To manage existing ESS recovery group pairs using **mmvdisk**, you must convert them to the **mmvdisk** IBM Storage Scale RAID configuration structure. This task is accomplished with the **mmvdisk recoverygroup convert** command.

The **mmvdisk recoverygroup list** command displays in the `remarks` column which existing recovery groups are not managed by **mmvdisk**:

```
# mmvdisk recoverygroup list

                                                          needs    user
recovery group  active   current or master server        service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
ESS01L          yes      ess01io1.gpfs.net                no            5  non-mmvdisk
ESS01R          yes      ess01io2.gpfs.net                no            5  non-mmvdisk
```

A recovery group pair must be converted together, and the two recovery groups must share an **mmvdisk** node class. An unused node class name must be specified, which will be created as an **mmvdisk** node class containing the two servers for the recovery group pair.

The IBM Storage Scale RAID configuration for the servers is set under the **mmvdisk** node class using the same settings that would be used for new **mmvdisk** recovery groups.

The user vdisk NSDs in the recovery group pair are matched into vdisk sets according to their file systems and attributes. The converted vdisk sets are given generated names, which can be changed later to something more meaningful using the **mmvdisk vdiskset rename** command.

To convert existing recovery groups ESS01L and ESS01R with a new **mmvdisk** node class ESS01, run the following command:

```
# mmvdisk recoverygroup convert --recovery-group ESS01L,ESS01R --node-class ESS01 --recycle one

mmvdisk: This command will permanently change the GNR configuration
mmvdisk: attributes and disable the legacy GNR command set for the
mmvdisk: servers and recovery groups involved, and their subsequent
mmvdisk: administration must be performed with the mmvdisk command.

mmvdisk: Do you wish to continue (yes or no)? yes

mmvdisk: Converting recovery groups 'ESS01L' and 'ESS01R'.
mmvdisk: Creating node class 'ESS01'.
mmvdisk: Adding 'ess01io1' to node class 'ESS01'.
mmvdisk: Adding 'ess01io2' to node class 'ESS01'.
mmvdisk: Associating recovery group 'ESS01L' with node class 'ESS01'.
mmvdisk: Associating recovery group 'ESS01R' with node class 'ESS01'.
mmvdisk: Recording pre-conversion cluster configuration in /var/mmfs/tmp/
mmvdisk.convert.ESS01L.ESS01R.before.m24
mmvdisk: Updating server configuration attributes.
```

```
mmvdisk: Checking resources for specified nodes.
mmvdisk: Setting configuration for node class 'ESS01'.
mmvdisk: Defining vdisk set 'VS001_fs3' with recovery group 'ESS01L' (vdisk 'ESS01Lv3fs3').
mmvdisk: Defining vdisk set 'VS002_fs4' with recovery group 'ESS01L' (vdisk 'ESS01Lv4fs4').
mmvdisk: Defining vdisk set 'VS003_fs1' with recovery group 'ESS01L' (vdisk 'ESS01Lv1fs1data').
mmvdisk: Defining vdisk set 'VS004_fs2' with recovery group 'ESS01L' (vdisk 'ESS01Lv2fs2').
mmvdisk: Defining vdisk set 'VS005_fs1' with recovery group 'ESS01L' (vdisk 'ESS01Lv1fs1meta').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS003_fs1' (vdisk 'ESS01Rv1fs1data').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS004_fs2' (vdisk 'ESS01Rv2fs2').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS001_fs3' (vdisk 'ESS01Rv3fs3').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS005_fs1' (vdisk 'ESS01Rv1fs1meta').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS002_fs4' (vdisk 'ESS01Rv4fs4').
mmvdisk: Committing cluster configuration changes.
mmvdisk: Recording post-conversion cluster configuration in /var/mmfs/tmp/
mmvdisk.convert.ESS01L.ESS01R.after.m24

mmvdisk: Restarting GPFS on the following nodes:
mmvdisk:     ess01io1.gpfs.net
mmvdisk: Restarting GPFS on the following nodes:
mmvdisk:     ess01io2.gpfs.net
```

The recovery groups ESS01L and ESS01R are now under **mmvdisk** management. The remarks column of **mmvdisk recoverygroup list** is blank:

```
# mmvdisk recoverygroup list

                                                          needs    user
recovery group  active   current or master server        service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
ESS01L          yes      ess01io1.gpfs.net                no            5
ESS01R          yes      ess01io2.gpfs.net                no            5
```

The converted vdisk sets can be listed to examine their attributes and sizing:

```
# mmvdisk vdiskset list --vdisk-set all

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
VS001_fs3           2   59 TiB   82 TiB  yes      fs3, DA1, 8+3p, 16 MiB, dataAndMetadata,
system
VS002_fs4           2  204 TiB  282 TiB  yes      fs4, DA1, 8+3p, 16 MiB, dataAndMetadata,
system
VS003_fs1           2  297 TiB  410 TiB  yes      fs1, DA1, 8+3p, 16 MiB, dataOnly, data
VS004_fs2           2   51 TiB   91 TiB  yes      fs2, DA1, 4+3p, 4 MiB, dataAndMetadata, system
VS005_fs1           2   11 TiB   45 TiB  yes      fs1, DA1, 4WayReplication, 1 MiB,
metadataOnly, system

                declustered           capacity           all vdisk sets defined
recovery group     array     total raw  free raw  free%  in the declustered array
--------------  -----------  ---------  --------  -----  ------------------------
ESS01L          DA1            911 TiB    64 GiB     0%  VS001_fs3, VS002_fs4, VS003_fs1,
VS004_fs2,
                                                          VS005_fs1
ESS01R          DA1            911 TiB    64 GiB     0%  VS001_fs3, VS002_fs4, VS003_fs1,
VS004_fs2,
                                                          VS005_fs1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  -----------------------
ESS01          29 GiB    16 GiB  VS001_fs3 (1448 MiB), VS002_fs4 (4994 MiB), VS003_fs1 (7249
MiB),
                                 VS004_fs2 (1921 MiB), VS005_fs1 (568 MiB)
```

The vdisk sets can be renamed to suit the administrator's preferences, and the **mmvdisk** file system characteristics can be listed:

```
# mmvdisk vdiskset rename --vdisk-set VS005_fs1 --new-name fs1meta
mmvdisk: Vdisk set 'VS005_fs1' renamed to 'fs1meta'.
# mmvdisk vdiskset rename --vdisk-set VS003_fs1 --new-name fs1data
mmvdisk: Vdisk set 'VS003_fs1' renamed to 'fs1data'.
# mmvdisk filesystem list --file-system fs1

                                vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
fs1data         ESS01L              1               1  no        yes    data
fs1data         ESS01R              1               2  no        yes    data
```

```
fs1meta         ESS01L             1              1  yes       no     system
fs1meta         ESS01R             1              2  yes       no     system
```

Multiple ESS building blocks, with multiple recovery group pairs, must be converted one pair at a time. If a pre-**mmvdisk** file system spans multiple recovery group pairs, it is not recognized as an **mmvdisk** file system until all of the recovery groups with vdisk NSDs for the file system are converted.

Replacing a pdisk using mmvdisk

If one or more pdisks in the recovery group is marked for replacement, the **mmvdisk recoverygroup list** command reports it with a `yes` in the `needs service` column.

In the following example, the BB01L recovery group needs service:

```
# mmvdisk recoverygroup list

                                                          needs    user
recovery group  active   current or master server        service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
BB01L           yes      server01.gpfs.net                yes           3
BB01R           yes      server02.gpfs.net                no            3
```

This happens when the number of failed pdisks in one of the recovery group's declustered arrays reaches or exceeds the replacement threshold for the declustered array.

Pdisks that have reached the threshold for replacement are listed with **mmvdisk pdisk list --replace**:

```
# mmvdisk pdisk list --recovery-group all --replace

recovery group  pdisk         priority  FRU (type)       location
--------------  ------------  --------  ---------------  --------
BB01L           e2s11             1.15  00W1240          Enclosure 2 Drive 11
BB01L           e3s01             1.15  00W1240          Enclosure 3 Drive 1

mmvdisk: A lower priority value means a higher need for replacement.
```

To replace the physical disk of a failed recovery group pdisk, complete the following tasks:

1. Prepare the pdisk for replacement.
2. Remove the failed physical disk.
3. Insert a new physical disk.
4. Replace the pdisk with the newly inserted physical disk.

To prepare pdisk e2s11 of recovery group BB01L for replacement, run the following command:

```
# mmvdisk pdisk replace --prepare --recovery-group BB01L --pdisk e2s11
mmvdisk: Suspending pdisk e2s11 of RG BB01L in location SX32901810-11.
mmvdisk: Location SX32901810-11 is Enclosure 2 Drive 11.
mmvdisk: Carrier released.
mmvdisk:
mmvdisk:   - Remove carrier.
mmvdisk:   - Replace disk in location SX32901810-11 with type '00W1240'.
mmvdisk:   - Reinsert carrier.
mmvdisk:   - Issue the following command:
mmvdisk:
mmvdisk:   mmvdisk pdisk replace --recovery-group BB01L --pdisk 'e2s11'
```

Then, remove the failed physical disk and insert a new physical disk of the same FRU/type.

Finish replacing pdisk e2s11 with the new physical disk by running the following command:

```
# mmvdisk pdisk replace --recovery-group BB01L --pdisk e2s11
mmvdisk:
mmvdisk: Preparing a new pdisk for use may take many minutes.
mmvdisk:
mmvdisk: The following pdisks will be formatted on node ess01io1:
mmvdisk:       /dev/sdrk
mmvdisk:
mmvdisk: Location SX32901810-11 is Enclosure 2 Drive 11.
```

```
mmvdisk: Pdisk e2s11 of RG BB01L successfully replaced.
mmvdisk: Carrier resumed.
```

Repeat this procedure for any other pdisk that is marked for replacement.

ESS MES upgrade

Beginning with ESS 5.3.2, it is possible to grow specific ESS building blocks by adding one or two enclosures. This is called an MES (miscellaneous equipment specification) upgrade, which is performed using the **mmvdisk recoverygroup resize** command.

Important: The MES upgrade of an ESS building block requires the participation of IBM Service because it involves the careful coordination of ESS hardware and IBM Storage Scale RAID software activity. This topic is limited to outlining how the **mmvdisk** command facilitates the IBM Storage Scale RAID part of the process. For more information about the integrated hardware and software aspects of an ESS MES upgrade, see *Deploying the Elastic Storage Server*.

The five permitted MES upgrade paths are described in MES upgrade paths:

Table 4. MES upgrade paths

Existing ESS building block	MES upgrade ESS building block	Enclosures added
GS1S	GS2S	One 5147-024 enclosure
GS2S	GS4S	Two 5147-024 enclosures
GL1S	GL2S	One 5147-084 enclosure
GL2S	GL4S	Two 5147-084 enclosures
GL4S	GL6S	Two 5147-084 enclosures

The new enclosures must be the same model as the existing enclosures and contain the same type and size disks.

Multiple MES upgrades must be performed in increments. For example, it is not possible to go directly from a GS1S to a GS4S; the intermediary MES upgrade of the GS1S to GS2S must be completed before the GS2S to GS4S MES upgrade can be performed.

The following example assumes that there is a GL2S building block with a nearly full file system (fs1) that uses all the GL2S capacity.

The vdisk set sizing for the file system vdisk set and the GL2S recovery groups shows no unclaimed space in the declustered arrays:

```
# mmvdisk vs list --vs fs1

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
fs1                 2  350 TiB  483 TiB  yes      fs1, DA1, 8+3p, 16 MiB, dataAndMetadata,
system


                declustered                 capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
BB03L           DA1          HDD     483 TiB    28 GiB     0%  fs1
BB03R           DA1          HDD     483 TiB    28 GiB     0%  fs1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB03          111 GiB  8925 MiB  fs1 (8540 MiB)
```

It is decided to double the size of the recovery groups with a GL2S to GL4S MES upgrade and to add the new capacity to the existing file system.

Before the new enclosures are connected and verified, the user declustered array DA1 in each of the GL2S recovery groups will show 83 pdisks and 2 pdisk-equivalent spare space:

```
# mmvdisk rg list --rg BB03L --da

declustered   needs    vdisks      pdisks     replace        capacity        scrub
   array     service  user log  total spare  threshold  total raw free raw  duration
background task
-----------  -------  ---- ---  ----- -----  ---------  --------- --------  --------
---------------
NVR          no          0   1      2     0          1          0        0   14 days  scrub
(84%)
SSD          no          0   1      1     0          1          0        0   14 days  scrub (4%)
DA1          no          1   1     83     2          2    483 TiB   28 GiB   14 days  scrub
(37%)

mmvdisk: Total capacity is the raw space before any vdisk set definitions.
mmvdisk: Free capacity is what remains for additional vdisk set definitions.
```

IBM Service installs the two new enclosures following the GL2S to GL4S MES upgrade procedure.

When the new enclosures have been verified to be correctly installed and configured on the servers, IBM service will perform the IBM Storage Scale RAID software part of the procedure.

First, the GL2S recovery groups must not have any failed disks:

```
# mmvdisk pdisk list --rg BB03L,BB03R --not-ok
mmvdisk: All pdisks of the specified recovery groups are ok.
```

The servers for the GL2S building block must now see a perfect match for the newly connected and upgraded GL4S disk topology:

```
# mmvdisk server list --disk-topology --nc BB03

 node                                    needs    matching
number  server                         attention   metric   disk topology
------  -------------------------------  ---------  --------  -------------
     1  server05.gpfs.net                no          100/100  ESS GL4S
     2  server06.gpfs.net                no          100/100  ESS GL4S
```

When the new building block configuration is seen perfectly by the servers, IBM Service will "resize" the recovery group pair to use the upgraded GL4S capacity:

```
# mmvdisk rg resize --rg BB03L,BB03R
mmvdisk: Obtaining pdisk information for recovery group 'BB03L'.
mmvdisk: Obtaining pdisk information for recovery group 'BB03R'.
mmvdisk: Analyzing disk topology for node 'server05.gpfs.net'.
mmvdisk: Analyzing disk topology for node 'server06.gpfs.net'.
mmvdisk: Validating existing pdisk locations for recovery group 'BB03L'.
mmvdisk: Validating existing pdisk locations for recovery group 'BB03R'.
mmvdisk: The resized server disk topology is 'ESS GL4S'.
mmvdisk: Validating declustered arrays for recovery group 'BB03L'.
mmvdisk: Validating declustered arrays for recovery group 'BB03R'.
mmvdisk: Adding new pdisks to recovery group 'BB03L'.
mmvdisk: Updating declustered array attributes for recovery group 'BB03L'.
mmvdisk: Adding new pdisks to recovery group 'BB03R'.
mmvdisk: Updating declustered array attributes for recovery group 'BB03R'.
mmvdisk: Successfully resized recovery groups 'BB03L' and 'BB03R'.
```

After the recovery group pair is resized, the user declustered array DA1 in each of the recovery groups has 167 pdisks with 4 pdisk-equivalent spare space, which is the standard for GL4S. IBM Storage Scale RAID also begins rebalancing vdisk data across the new DA1 capacity:

```
# mmvdisk rg list --rg BB03L --da

declustered   needs    vdisks      pdisks     replace        capacity        scrub
   array     service  user log  total spare  threshold  total raw free raw  duration
background task
-----------  -------  ---- ---  ----- -----  ---------  --------- --------  --------
---------------
```

```
NVR          no          0   1      2     0          1          0        0   14 days  scrub
(88%)
SSD          no          0   1      1     0          1          0        0   14 days  scrub (4%)
DA1          no          1   1    167     4          2    972 TiB  489 TiB   14 days  rebalance
(0%)

mmvdisk: Total capacity is the raw space before any vdisk set definitions.
mmvdisk: Free capacity is what remains for additional vdisk set definitions.
```

The doubling in size of the recovery group pair means that there is now 50% free capacity for the creation of additional vdisk sets to increase the size of file system fs1:

```
# mmvdisk vs list --vs all

                    member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
fs1                 2  350 TiB  483 TiB  yes      fs1, DA1, 8+3p, 16 MiB, dataAndMetadata,
system

                declustered                 capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
BB03L           DA1          HDD     972 TiB   489 TiB    50%  fs1
BB03R           DA1          HDD     972 TiB   489 TiB    50%  fs1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB03          111 GiB  8925 MiB  fs1 (8540 MiB)
```

To double the size of the file system, a new vdisk set can be defined, created, and added to the file system. The **mmvdisk vdiskset define** command has an option to define a new vdisk set by copying the definition of an existing vdisk set. This ensures that the new vdisk set will have exactly the same size vdisk NSDs as those already in the file system.

To prepare for making a copy of a vdisk set, first the existing vdisk set is renamed from fs1 to fs1.vs1:

```
# mmvdisk vs rename --vs fs1 --new-name fs1.vs1
mmvdisk: Vdisk set 'fs1' renamed to 'fs1.vs1'.
```

Copying a vdisk set definition checks that the declustered array for the copy is compatible with the declustered array from the original definition. This leads to the paradoxical situation where the attempt to copy vdisk set fs1.vs1 to a new vdisk set fs1.vs2 in the same recovery groups and declustered arrays is found to be incompatible:

```
# mmvdisk vs define --vs fs1.vs2 --copy fs1.vs1
mmvdisk: Declustered array 'DA1' in recovery group 'BB03L' is incompatible with
mmvdisk: recovery group 'BB03L' in vdisk set 'fs1.vs1' because:
mmvdisk:   Different pdisks per server in the declustered array, 167 versus 83.
mmvdisk: Declustered array 'DA1' in recovery group 'BB03R' is incompatible with
mmvdisk: recovery group 'BB03L' in vdisk set 'fs1.vs1' because:
mmvdisk:   Different pdisks per server in the declustered array, 167 versus 83.
mmvdisk: Command failed. Examine previous error messages to determine cause.
```

This is because the original vdisk set was defined when there were only 83 pdisks in DA1, and the copy is being made after DA1 was resized to 167 pdisks. The **--force-incompatible** flag may be used to allow the copy to be made even though the number of pdisks is now different:

```
# mmvdisk vs define --vs fs1.vs2 --copy fs1.vs1 --force-incompatible
mmvdisk: Vdisk set 'fs1.vs2' has been defined.
mmvdisk: Recovery group 'BB03L' has been defined in vdisk set 'fs1.vs2'.
mmvdisk: Recovery group 'BB03R' has been defined in vdisk set 'fs1.vs2'.

                    member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
fs1.vs2             2  350 TiB  483 TiB  no       -, DA1, 8+3p, 16 MiB, dataAndMetadata, system

                declustered                 capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
BB03L           DA1          HDD     972 TiB  6533 GiB     0%  fs1.vs1, fs1.vs2
```

```
BB03R           DA1          HDD     972 TiB  6533 GiB     0%  fs1.vs1, fs1.vs2

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
BB03          111 GiB    17 GiB  fs1.vs1 (8540 MiB), fs1.vs2 (8540 MiB)
```

Together, the original vdisk set fs1.vs1 and the copied vdisk set fs1.vs2 use all the space in the resized declustered arrays.

The copied vdisk set can then be created and added to file system fs1, doubling its size:

```
# mmvdisk vs create --vs fs1.vs2
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'fs1.vs2'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG005VS002
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG006VS002
mmvdisk: Created all vdisks in vdisk set 'fs1.vs2'.
mmvdisk: (mmcrnsd) Processing disk RG005VS002
mmvdisk: (mmcrnsd) Processing disk RG006VS002
mmvdisk: Created all NSDs in vdisk set 'fs1.vs2'.
# mmvdisk fs add --fs fs1 --vs fs1.vs2
mmvdisk: The following disks of fs1 will be formatted on node fsmgr01:
mmvdisk:     RG005VS002: size 367003904 MB
mmvdisk:     RG006VS002: size 367003904 MB
mmvdisk: Extending Allocation Map
mmvdisk: Checking Allocation Map for storage pool system
mmvdisk:   47 % complete on Tue Nov  6 22:22:27 2018
mmvdisk:   51 % complete on Tue Nov  6 22:22:32 2018
mmvdisk:   94 % complete on Tue Nov  6 22:22:37 2018
mmvdisk:  100 % complete on Tue Nov  6 22:22:37 2018
mmvdisk: Completed adding disks to file system fs1.
# mmvdisk fs list --fs fs1

                                 vdisk     list of       holds    holds  storage
vdisk set       recovery group   count  failure groups  metadata  data    pool
--------------  --------------   -----  --------------  --------  -----  -------
fs1.vs1         BB03L               1               1  yes       yes    system
fs1.vs1         BB03R               1               2  yes       yes    system
fs1.vs2         BB03L               1               1  yes       yes    system
fs1.vs2         BB03R               1               2  yes       yes    system

#
```

Limitations of mmvdisk

The **mmvdisk** command does not integrate these IBM Storage Scale RAID features:

- Component database management
- Firmware management

Note: These features must continue to be managed using the component database management commands (for example, **mmlscomp**) and firmware management commands (for example, **mmlsfirmware**).

Chapter 4. Managing IBM Storage Scale RAID

This section includes information about the overall management of IBM Storage Scale RAID. It also describes, in more detail, the characteristics and behaviors of these IBM Storage Scale RAID concepts and entities: declustered arrays, recovery groups, pdisks, pdisk-group fault tolerance, and vdisks.

You can use the ESS GUI to perform various IBM Storage Scale RAID management tasks.

Recovery groups

A *recovery group* is the fundamental organizing structure employed by IBM Storage Scale RAID. A recovery group is conceptually the internal GPFS equivalent of a hardware disk controller. Within a recovery group, individual JBOD disks are defined as *pdisks* and assigned to *declustered arrays*. Each pdisk belongs to exactly one declustered array within one recovery group. Within a declustered array of pdisks, *vdisks* are defined. The vdisks are the equivalent of the RAID logical unit numbers (LUNs) for a hardware disk controller. One or two GPFS cluster nodes must be defined as the servers for a recovery group, and these servers must have direct hardware connections to the JBOD disks in the recovery group. Two servers are recommended for high availability server failover, but only one server will actively manage the recovery group at any given time. One server is the preferred and *primary* server, and the other server, if defined, is the *backup* server.

Multiple recovery groups can be defined, and a GPFS cluster node can be the primary or backup server for more than one recovery group. The name of a recovery group must be unique within a GPFS cluster.

Recovery group server parameters

To enable a GPFS cluster node as a recovery group server, it must have the **mmchconfig** configuration parameter **nsdRAIDTracks** set to a nonzero value, and the GPFS daemon must be restarted on the node. The **nsdRAIDTracks** parameter defines the maximum number of vdisk track descriptors that the server can have in memory at a given time. The volume of actual vdisk data that the server can cache in memory is governed by the size of the GPFS page pool on the server and the value of the **nsdRAIDBufferPoolSizePct** configuration parameter. The **nsdRAIDBufferPoolSizePct** parameter defaults to 80% of the page pool on the server. A recovery group server should be configured with a substantial amount of page pool, on the order of tens of gigabytes. A recovery group server becomes an NSD server after NSDs are defined on the vdisks in the recovery group, so the **nsdBufSpace** parameter also applies. The default for **nsdBufSpace** is 30% of the page pool, and it can be decreased to its minimum value of 10% because the vdisk data buffer pool is used directly to serve the vdisk NSDs.

The vdisk track descriptors, as governed by **nsdRAIDTracks**, include such information as the RAID code, track number, and status. The descriptors also contain pointers to vdisk data buffers in the GPFS page pool, as governed by **nsdRAIDBufferPoolSizePct**. It is these buffers that hold the actual vdisk data and redundancy information.

For more information on how to set the **nsdRAIDTracks** and **nsdRAIDBufferPoolSizePct** parameters, see Appendix A, “Best-practice recommendations for IBM Storage Scale RAID,” on page 259.

For more information on the **nsdRAIDTracks**, **nsdRAIDBufferPoolSizePct**, and **nsdBufSpace** parameters, see the **mmchconfig** command in the *IBM Storage Scale: Command and Programming Reference*.

Recovery group creation

Recovery groups are created using the **mmcrrecoverygroup** command, which takes the following arguments:

- The name of the recovery group to create.

- The name of a stanza file describing the declustered arrays and pdisks within the recovery group.
- The names of the GPFS cluster nodes that will be the primary and, if specified, backup servers for the recovery group.

When a recovery group is created, the GPFS daemon must be running with the **nsdRAIDTracks** configuration parameter in effect on the specified servers.

See the "mmcrrecoverygroup command" on page 288 for more information.

Recovery group server failover

When, as is recommended, a recovery group is assigned two servers, one server is the preferred and *primary* server for the recovery group and the other server is the *backup* server. Only one server can serve the recovery group at any given time; this server is known as the *active* recovery group server. The server that is not currently serving the recovery group is the *standby* server. If the active recovery group server is unable to serve a recovery group, it will relinquish control of the recovery group and pass it to the standby server, if available. The failover from the active to the standby server should be transparent to any GPFS file system using the vdisk NSDs in the recovery group. There will be a pause in access to the file system data in the vdisk NSDs of the recovery group while the recovery operation takes place on the new server. This server failover recovery operation involves the new server opening the component disks of the recovery group and playing back any logged RAID transactions.

The active server for a recovery group can be changed by the IBM Storage Scale RAID administrator using the `mmchrecoverygroup` command. For planned temporary recovery group reassignment use the `mmchrecoverygroup` command to change the active server for maintenance operations. This command can also be used to change the primary and backup servers for a recovery group.

See the "mmchrecoverygroup command" on page 285 for more information.

Pdisks

The IBM Storage Scale RAID *pdisk* is an abstraction of a physical disk. A pdisk corresponds to exactly one physical disk and belongs to exactly one declustered array within exactly one recovery group. Before discussing how declustered arrays collect pdisks into groups, it will be useful to describe the characteristics of pdisks.

A recovery group can contain a maximum of 512 pdisks. A declustered array within a recovery group can contain a maximum of 512 pdisks. The name of a pdisk must be unique within a recovery group; that is, two recovery groups can each contain a pdisk named `disk10`, but a recovery group cannot contain two pdisks named `disk10`, even if they are in different declustered arrays.

A pdisk is usually created using the **mmcrrecoverygroup** command, whereby it is assigned to a declustered array within a newly created recovery group. In unusual situations, pdisks can also be created and assigned to a declustered array of an existing recovery group by using the **mmaddpdisk** command.

To create a pdisk, a stanza must be supplied to the **mmcrrecoverygroup** or **mmaddpdisk** commands specifying the pdisk name, the declustered array name to which it is assigned, and a block device special file name for the entire physical disk as it is configured by the operating system on the active recovery group server. A sample pdisk creation stanza follows:

```
%pdisk: pdiskName=c073d1
        device=/dev/hdisk192
        da=DA1
        nPathActive=2
        nPathTotal=4
```

Other stanza parameters might be present. For more information about pdisk stanza parameters, see "Pdisk stanza format" on page 55.

The device name for a pdisk must refer to the entirety of a single physical disk; pdisks should not be created using virtualized or software-based disks (for example, logical volumes, disk partitions, logical

units from other RAID controllers, or network-attached disks). The exception to this rule are non-volatile RAM (NVRAM) volumes used for the log tip vdisk, which is described in "Log vdisks" on page 62. For a pdisk to be created successfully, the physical disk must be present and functional at the specified device name on the active server. The physical disk must also be present on the standby recovery group server, if one is configured. The physical disk block device special name on the standby server will almost certainly be different and will be discovered automatically by IBM Storage Scale.

The attributes of a pdisk include the physical disk's unique worldwide name (WWN), its field replaceable unit (FRU) code, and its physical location code. Pdisk attributes can be displayed using the **mmlspdisk** command; of particular interest here are the pdisk device *paths* and the pdisk *states*.

Pdisks that have failed and have been marked for replacement by the disk hospital are replaced using the **mmchcarrier** command. In unusual situations, pdisks can be added or deleted using the **mmaddpdisk** or **mmdelpdisk** commands. When deleted, either through replacement or the **mmdelpdisk** command, the pdisk abstraction will only cease to exist when all of the data it contained has been rebuilt onto spare space (even though the physical disk might have been removed from the system).

Pdisks are normally under the control of IBM Storage Scale RAID and the disk hospital. In some situations, however, the **mmchpdisk** command can be used to manipulate pdisks directly. For example, if a pdisk has to be removed temporarily to allow for hardware maintenance on other parts of the system, you can use the **mmchpdisk --begin-service-drain** command to drain the data before removing the pdisk. After bringing the pdisk back online, you can use the **mmchpdisk --end-service-drain** command to return the drained data to the pdisk.

Note: This process requires that there be sufficient spare space in the declustered array for the data that is to be drained. If the available spare space is insufficient, it can be increased with the **mmchrecoverygroup** command.

Pdisk paths

To the operating system, physical disks are made visible as block devices with device special file names, such as `/dev/sdbc` (on Linux) or `/dev/hdisk32` (on AIX®). Most pdisks that IBM Storage Scale RAID uses are located in JBOD arrays, except for the NVRAM pdisk that is used for the log tip vdisk. To achieve high availability and throughput, the physical disks of a JBOD array are connected to each server by multiple (usually two) interfaces in a configuration known as *multipath* (or *dualpath*). When two operating system block devices are visible for each physical disk, IBM Storage Scale RAID refers to them as the *paths* to the pdisk.

In normal operation, the paths to individual pdisks are discovered by IBM Storage Scale RAID automatically. There are only two instances when a pdisk must be referred to by its explicit block device path name: during recovery group creation using the **mmcrrecoverygroup** command, and when adding new pdisks to an existing recovery group with the **mmaddpdisk** command. In both of these cases, only one of the block device path names as seen on the active server needs to be specified; any other paths on the active and standby servers will be discovered automatically. For each pdisk, the `nPathActive` and `nPathTotal` stanza parameters can be used to specify the expected number of paths to that pdisk, from the active server and from all servers. This allows the disk hospital to verify that all expected paths are present and functioning.

The operating system could have the ability to internally merge multiple paths to a physical disk into a single block device. When IBM Storage Scale RAID is in use, the operating system multipath merge function must be disabled because IBM Storage Scale RAID itself manages the individual paths to the disk. For more information, see "Configuring GNR recovery groups: a sample scenario" on page 419.

Pdisk stanza format

Pdisk stanzas have three mandatory parameters and six optional parameters, and they look like this:

```
%pdisk: pdiskName=PdiskName
        device=BlockDeviceName
        da=DeclusteredArrayName
```

```
        [nPathActive=ExpectedNumberActivePaths]
        [nPathTotal=ExpectedNumberTotalPaths]
        [rotationRate=HardwareRotationRate]
        [fruNumber=FieldReplaceableUnitNumber]
        [location=PdiskLocation]
        [nsdFormatVersion=DesiredNsdFormatVersion]
```

where:

pdiskName=*PdiskName*
Specifies the name of a pdisk.

device=*BlockDeviceName*
Specifies the name of a block device. The value provided for *BlockDeviceName* must refer to the block device as configured by the operating system on the primary recovery group server or have the node name prefixed to the device block name.

Sample values for *BlockDeviceName* are `/dev/sdbc` and `//nodename/dev/sdbc` (on Linux), and `hdisk32`, `/dev/hdisk32` and `//nodename/dev/hdisk32` (on AIX).

Only one *BlockDeviceName* needs to be used, even if the device uses multipath and has multiple device names.

da=*DeclusteredArrayName*
Specifies the *DeclusteredArrayName* in the pdisk stanza, which implicitly creates the declustered array with default parameters.

nPathActive=*ExpectedNumberActivePaths*
Specifies the expected number of paths for the connection from the active server to this pdisk. If this parameter is specified, the `mmlsrecoverygroup` and `mmlspdisk` commands will display warnings if the number of paths does not match the expected number for a pdisk that should be functioning normally. If this parameter is not specified, the default is 0, which means "do not issue such warnings".

Sample values are 2 for all pdisks that are located in an ESS disk enclosure (or the IBM Power 775 Disk Enclosure) and 1 for the NVRAM pdisk that is used for the log tip vdisk.

nPathTotal=*ExpectedNumberTotalPaths*
Specifies the expected number of paths for the connection from all active and backup servers to this pdisk. If this parameter is specified, the `mmlsrecoverygroup` and `mmlspdisk` commands will display warnings if the number of paths does not match the expected number, for a pdisk that should be functioning normally. If this parameter is not specified, the default is 0, which means "do not issue such warnings".

Sample values are 4 for all pdisks located in an ESS disk enclosure (or the IBM Power 775 Disk Enclosure) and 1 for the NVRAM pdisk used for the log tip vdisk.

rotationRate=*HardwareRotationRate*
Specifies the hardware type of the pdisk: NVRAM, SSD, or a rotating HDD. The only valid values are the string `NVRAM`, the string `SSD`, or a number between 1025 and 65535 (inclusive) indicating the rotation rate in revolutions per minute for HDDs. For all pdisks that are used in an ESS disk enclosure (or the IBM Power 775 Disk Enclosure), there is no need to specify this parameter, as the hardware type and rotation rate will be determined from the hardware automatically. This parameter should only be specified for the NVRAM pdisk on the ESS. The default is to rely on the hardware to identify itself, or leave the hardware type and rotation rate unknown if the hardware does not have the ability to identify itself.

A sample value is the string `NVRAM` for the NVRAM pdisk used for the log tip vdisk.

fruNumber=*FieldReplaceableUnitNumber*
Specifies the unit number for the field replaceable unit that is needed to repair this pdisk if it fails. For all pdisks used in an ESS disk enclosure (or the IBM Power 775 Disk Enclosure), there is no need to specify this parameter, as it is automatically determined from the hardware. For the NVRAM pdisk used in the log tip vdisk, the user can enter a string here, which will be displayed to service personnel when replacement of that pdisk is performed. However, setting this value for the NVRAM pdisk is not required, as the service replacement procedure for that pdisk is specific to that particular

type of hardware. The default is to rely on the hardware to identify itself, or to leave the FRU number unknown if the hardware does not have the ability to identify itself.

location=*PdiskLocation*
Specifies the physical location of this pdisk. For all pdisks used in an ESS disk enclosure (or the IBM Power 775 Disk Enclosure), there is no need to specify this parameter, as it automatically determined from the hardware. For the NVRAM pdisk used in the log tip vdisk, the user can enter a string here, which will be displayed in the output of mmlspdisk. The default is to rely on the location reported by the hardware, or leave the location unknown.

A sample value is SV21314035-5-1, which describes a pdisk in enclosure serial number SV21314035, drawer 5, slot 1.

nsdFormatVersion=*DesiredNsdFormatVersion*
Specifies the desired Nsd Format version for this pdisk. The value can be either 1 or 2. This parameter is only effective with recovery group version 4.2.0.1 or later. If this parameter is not specified, the pdisk Nsd version will be 2 for recovery group version 4.2.0.1 or later. For recovery group version 4.1.0.1 or earlier, the Nsd version can only be 1.

Pdisk states

IBM Storage Scale RAID maintains its view of a pdisk and its corresponding physical disk by using a *pdisk state*. The pdisk state consists of multiple keyword flags, which can be displayed by using the **mmlsrecoverygroup** or **mmlspdisk** commands. You can also use the ESS GUI to display pdisk states. The state of pdisks is displayed in these views: **Arrays > Physical**, **Monitoring > System**, and **Monitoring > System Details**. In addition, information about pdisks with a negative state (disks that should be replaced, for example) is displayed in the **Monitoring > Events** view.

The pdisk state flags indicate in detail how IBM Storage Scale RAID is using or managing a disk.

In normal circumstances, the state of most of the pdisks is represented by the sole keyword ok. The ok keyword means that IBM Storage Scale RAID considers the pdisk to be healthy: the recovery group server is able to communicate with the disk, the disk is functioning normally, and the disk can be used to store data. The `diagnosing` flag is present in the pdisk state when the IBM Storage Scale RAID disk hospital suspects, or attempts to correct, a problem. If IBM Storage Scale RAID is unable to communicate with a disk, the pdisk state includes the `missing` keyword. If a `missing` disk becomes reconnected and functions properly, its state changes back to ok. The `readonly` flag means that a disk is indicating that it can no longer safely write data. A disk can also be marked by the disk hospital as `failing`, which might be due to an excessive number of media or checksum errors. When the disk hospital concludes that a disk is no longer operating effectively, it declares the disk `dead`. If the number of non-functioning (`dead`, `missing`, `failing`, or `slow`) pdisks reaches or exceeds the replacement threshold of their declustered array, the disk hospital adds the `replace` flag to the pdisk state, which indicates that physical disk replacement should be performed as soon as possible.

When the state of a pdisk indicates that it can no longer behave reliably, IBM Storage Scale RAID rebuilds the pdisk's data onto spare space on the other pdisks in the same declustered array. This is called *draining* the pdisk. Flags indicate whether a pdisk is draining or was drained. The `draining` flag means that IBM Storage Scale RAID will rebuild the data from the pdisk. The `deleting` flag means that the IBM Storage Scale RAID administrator issued the **mmdelpdisk** command to delete the pdisk.

To summarize, most of the pdisks are in the ok state during normal operation. The ok state indicates that the disk is reachable, functioning, not draining, and that the disk contains user data as well as IBM Storage Scale RAID recovery group and vdisk configuration information. A more complex example of a pdisk state is `dead/drained` for a single pdisk that has failed. This set of pdisk state flags indicates that the pdisk was declared dead by the system, was marked to be drained, and that all of its data (recovery group, vdisk configuration, and user) was successfully rebuilt onto the spare space on other pdisks.

In addition to the states discussed here, there are some transient pdisk states that have little impact on normal operations. Table 5 on page 58 lists the complete set of states.

Table 5. Pdisk states

State	Description
ok	The disk is available.
dead	The disk completely failed.
simulatedDead	The disk is being treated as if it were dead for error injection (see **mmchpdisk --simulate-dead**).
missing	The disk hospital determined that the system cannot connect to the drive.
readonly	The disk has failed; it can still be read but not written.
failing	The disk needs to be drained and replaced due to a SMART trip or high uncorrectable error rate.
simulatedFailing	The disk is being treated as if it were failing for error injection (see **mmchpdisk --simulate-failing**).
slow	The disk needs to be drained and replaced due to poor performance.
diagnosing	The disk hospital is checking the disk after an error.
PTOW	The disk is temporarily unavailable because of a pending timed-out write.
suspended	The disk is temporarily offline for service (see **mmchpdisk** and **mmchcarrier**).
serviceDrain	The disk is being drained of data for service (see **mmchpdisk --begin-service-drain**).
draining	The data is being drained from the disk and moved to distributed spare space on other disks.
deleting	The disk is being deleted from the system through the **mmdelpdisk**, **mmaddpdisk/--replace**, or **mmchcarrier** command.
drained	All of the data was successfully drained from the disk and the disk is replaceable, but the replace threshold was not met.
undrainable	As much of the data as possible was drained from the disk and moved to distributed spare space.
replace	The disk is ready for replacement.

Related information

- “mmdelpdisk command” on page 299
- “mmlspdisk command” on page 324
- “mmlsrecoverygroup command” on page 327

Declustered arrays

Note: The features available for actual declustered arrays will depend on the specific supported disk configuration in an IBM Storage Scale RAID installation.

Declustered arrays are disjoint subsets of the pdisks in a recovery group. Vdisks are created within declustered arrays, and vdisk tracks are declustered across all of an array's pdisks. The number of declustered arrays in a recovery group is determined by the enclosure and disk hardware configuration of the IBM Storage Scale RAID server for the recovery group. Depending on the specific configuration, an individual declustered array may contain up to 512 pdisks; however, the total number of pdisks in all declustered arrays within a recovery group may not exceed 512. A pdisk can belong to only one declustered array. The name of a declustered array must be unique within a recovery group; that is, two

recovery groups can each contain a declustered array named DA3, but a recovery group cannot contain two declustered arrays named DA3. The pdisks within a declustered array must all be of the same size and should all have similar performance characteristics.

A declustered array is usually created together with its member pdisks and its containing recovery group through the use of the **mmcrrecoverygroup** command. A declustered array can also be created using the **mmaddpdisk** command to add pdisks to a declustered array that does not yet exist in a recovery group. A declustered array may be deleted by deleting its last member pdisk, or by deleting the recovery group in which it resides. Any vdisk NSDs and vdisks within the declustered array must already have been deleted. There are no explicit commands to create or delete declustered arrays.

The main purpose of a declustered array is to group physical disks that are similar in performance characteristics and use. As vdisks are contained within a single declustered array, mixing pdisks of varying performance within a declustered array would not use the disks optimally. In a typical IBM Storage Scale RAID system, the first declustered array contains SSD pdisks that are used for the log vdisk, or the log backup vdisk if configured. If the system is configured to use a log tip vdisk (see "Log vdisks" on page 62), another declustered array contains NVRAM pdisks for that vdisk. Vdisks that are GPFS NSDs are then contained in one or more declustered arrays using high-capacity HDDs or SSDs.

A secondary purpose of declustered arrays is to partition disks that share a common point of failure or unavailability, such as removable carriers that hold multiple disks. This comes into play when one considers that removing a multi-disk carrier to perform disk replacement also temporarily removes some good disks, perhaps a number in excess of the fault tolerance of the vdisk NSDs. This would cause temporary suspension of file system activity until the disks are restored. To avoid this, each disk position in a removable carrier should be used to define a separate declustered array, such that disk position one defines DA1, disk position two defines DA2, and so on. Then when a disk carrier is removed, each declustered array will suffer the loss of just one disk, which is within the fault tolerance of any IBM Storage Scale RAID vdisk NSD.

The Declustered Array dashboard in the Elastic Storage Server GUI displays the IBM FlashCore Module (FCM) specified space and health information that is provided by the ZIMon collector. It displays the historical physical space utilization per Declustered Array (DA) under the following categories:

- Total capacity - The sum of the capacity of all pdisks in the DA.
- Physical capacity - The sum of the physical capacity of all pdisks in the DA.
- Used capacity - The sum of the used capacity of all pdisks in the DA.
- Used Physical capacity - The used physical capacity of the DA that is calculated using the used physical capacity of the most used pdisk.

Declustered array parameters

Declustered arrays have four parameters that can be set using stanza parameters when creating a declustered array, and can be changed using the **mmchrecoverygroup** command with the **--declustered-array** option. These are:

dataSpares
The number of disks' worth of equivalent spare space used for rebuilding vdisk data if pdisks fail. This defaults to one for arrays with nine or fewer pdisks, and two for arrays with 10 or more pdisks.

vcdSpares
The number of disks that can be unavailable while the IBM Storage Scale RAID server continues to function with full replication of vdisk configuration data (VCD). This value defaults to the number of data spares. To enable pdisk-group fault tolerance, this parameter is typically set to a larger value during initial system configuration (half of the number of pdisks in the declustered array + 1, for example).

replaceThreshold
The number of disks that must fail before the declustered array is marked as needing to have disks replaced. The default is the number of data spares.

scrubDuration
The number of days over which all the vdisks in the declustered array are scrubbed for errors. The default is 14 days.

Declustered array size

IBM Storage Scale RAID distinguishes between large and small declustered arrays. A declustered array is considered *large* if, at the time of its creation, the number of its pdisks is at least the setting of the `vcdSpares` parameter + 9. When using the default values for the `vcdSpares` and `dataSpares` parameters, this means that a declustered array is considered large if it contains at least 11 pdisks. All other declustered arrays are considered *small*. At least one declustered array in each recovery group must be large, because only large declustered arrays have enough pdisks to safely store an adequate number of replicas of the IBM Storage Scale RAID configuration data for the recovery group.

Because the narrowest RAID code that IBM Storage Scale RAID supports for user data is 3-way replication, the smallest possible declustered array contains four pdisks, including the minimum required equivalent spare space of one disk. The RAID code width of the intended vdisk NSDs and the amount of equivalent spare space also affect declustered array size; if Reed-Solomon 8 + 3p vdisks, which have a code width of 11, are required, and two disks of equivalent spare space is also required, the declustered array must have at least 13 member pdisks. Declustered arrays that contain only log vdisks can be smaller than these limits.

Data spare space and VCD spares

While operating with a failed pdisk in a declustered array, IBM Storage Scale RAID continues to serve file system I/O requests by using redundancy information on other pdisks to reconstruct data that cannot be read, and by marking data that cannot be written to the failed pdisk as stale. Meanwhile, to restore full redundancy and fault tolerance, the data on the failed pdisk is rebuilt onto *data spare space*, reserved unused portions of the declustered array that are declustered over all of the member pdisks. The failed disk is thereby *drained* of its data by copying it to the data spare space.

The amount of data spare space in a declustered array is set at creation time and can be changed later. The data spare space is expressed in whole units equivalent to the capacity of a member pdisk of the declustered array, but is spread among all of the member pdisks. There are no dedicated spare pdisks. This implies that a number of pdisks equal to the specified data spare space could fail, and the full redundancy of all of the data in the declustered array can be restored through a rebuild operation. If the user chooses to not fill the space in the declustered array with vdisks, and wants to use the unallocated space as extra data spare space, the user can increase the setting of the `dataSpares` parameter to the desired level of resilience against pdisk failures.

At a minimum, each declustered array normally requires data spare space that is equivalent to the size of one member pdisk. The exceptions, which have zero data spares and zero VCD spares, are declustered arrays that consist of the following:

- Non-volatile RAM disks used for a log tip vdisk
- SSDs used for a log tip backup vdisk.

Because large declustered arrays have a greater probability of disk failure, the default amount of data spare space depends on the size of the declustered array. A declustered array with nine or fewer pdisks defaults to having one disk of equivalent data spare space. A declustered array with 10 or more disks defaults to having two disks of equivalent data spare space. These defaults can be overridden, especially at declustered array creation. However, if at a later point too much of the declustered array is already allocated for use by vdisks, it might not be possible to increase the amount of data spare space.

IBM Storage Scale RAID *vdisk configuration data (VCD)* is stored more redundantly than vdisk content, typically 5-way replicated. When a pdisk fails, this configuration data is rebuilt at the highest priority, onto functioning pdisks. The redundancy of configuration data always has to be maintained, and IBM Storage Scale RAID will not serve a declustered array that does not have sufficient pdisks to store all configuration data at full redundancy. The declustered array parameter `vcdSpares` determines how many pdisks can fail and have full VCD redundancy restored, by reserving room on each pdisk for vdisk configuration data.

When using pdisk-group fault tolerance, the value of `vcdSpares` should be set higher than the value of the `dataSpares` parameter to account for the expected failure of hardware failure domains.

Increasing VCD spares

When new recovery groups are created, the **mkrginput** script sets recommended values for VCD spares.

To increase the VCD spares for existing recovery groups, use the **mmchrecoverygroup** command. See the "mmchrecoverygroup command" on page 285 for more information.

Declustered array free space

The declustered array free space reported by the **mmlsrecoverygroup** command reflects the space available for creating vdisks. Spare space is not included in this value since it is not available for creating new vdisks.

Pdisk free space

The pdisk free space reported by the **mmlsrecoverygroup** command reflects the actual number of unused data partitions on the disk. This includes spare space, so if a pdisk fails, these values will decrease as data is moved to the spare space.

Vdisks

Vdisks are created across the pdisks within a declustered array. Each recovery group requires a special *log home vdisk* to function (along with other log-type vdisks, as appropriate for specific environments); see "Log vdisks" on page 62. All other vdisks are created for use as GPFS file system NSDs.

A recovery group can contain at most 512 vdisks including log vdisks. Vdisks can be allocated arbitrarily among declustered arrays. Vdisks are created with the **mmcrvdisk** command. The **mmdelvdisk** command destroys vdisks and all their contained data.

When creating a vdisk, you must specify the RAID code, block size, vdisk size, and a name that is unique within the recovery group and the GPFS cluster. There are no adjustable parameters available for vdisks.

RAID code

The type, performance, and space efficiency of the RAID codes that are used for vdisks, discussed in "RAID codes" on page 2, should be considered when choosing the RAID code for a particular set of user data. GPFS storage pools and policy-based data placement can be used to ensure that data is stored with appropriate RAID codes.

If an IBM Storage Scale RAID server cannot serve a vdisk due to too many unreachable pdisks (pdisks in the missing state), the server fails over the recovery group to the backup server in hopes that the backup server has better I/O connectivity. If the backup server cannot serve vdisks either due to too many missing pdisks, it fails the recovery group back to the primary server, which repeats the cycle until I/O connectivity improves. This approach is aimed at making the system robust to temporary pdisk connectivity problems by allowing the administrator to restore pdisk connectivity before the system fails clients I/Os due to missing pdisks.

If the problem is caused by too many pdisks in other non-functioning states (for example, dead state), the recovery group will not failover to the partner node because the pdisk state is not expected to improve. It is expected to remain in the same state even if tried by the partner node. In this case, the client I/Os that are affected by too many such pdisks will fail.

A recovery group can contain vdisks of different levels of fault tolerances. In this case, the recovery group might contain some vdisks with sufficient fault tolerance that could be served despite the missing pdisks. Nevertheless, the vdisks might fail over to the partner server because the unit of failover is a recovery group with all its associated vdisks. Mixing vdisks of varying fault tolerance levels in the same recovery

group is allowed; however, in the presence of too many unreachable pdisks, service of vdisks with higher fault tolerance might be limited by vdisks with lower fault tolerance.

Block size

The vdisk block size must equal the GPFS file system block size of the storage pool where the vdisk is assigned. For replication codes, the supported vdisk block sizes are 256 KiB, 512 KiB, 1 MiB, and 2 MiB. For Reed-Solomon codes, the supported vdisk block sizes are 512 KiB, 1 MiB, 2 MiB, 4 MiB, 8 MiB, and 16 MiB. See Appendix A, “Best-practice recommendations for IBM Storage Scale RAID,” on page 259 for some vdisk configuration considerations.

The **maxblocksize** configuration attribute of the IBM Storage Scale **mmchconfig** command must be set appropriately for all nodes. The value of **maxblocksize** must be greater than or equal to the maximum block size of the vdisks. For more information about this attribute, see the **mmchconfig** command description in the *IBM Storage Scale: Command and Programming Reference*.

Vdisk size

The maximum vdisk size is the total space available on the pdisks in the declustered array, taking into account the overhead of the RAID code, minus spare space, minus vdisk configuration data, and minus a small amount of space reserved as a buffer for write operations. IBM Storage Scale RAID will round up the requested vdisk size as required. When creating a vdisk, the user can specify to use all remaining space in the declustered array for that vdisk.

Log vdisks

IBM Storage Scale RAID uses log vdisks to store such internal information as event log entries, updates to vdisk configuration data, and certain data write operations quickly. There are four types of log vdisks, as follows. Among them, they can be created and destroyed in any order.

log home vdisk
Every recovery group requires one log home vdisk to function. The log home vdisk must be created before any other non-log vdisks in the recovery group, and it can only be deleted after all other non-log vdisks in the recovery group have been deleted. The log home vdisk is divided into four sublogs: long-term event log, short-term event log, metadata log, and fast-write log to log small write operations.

log tip vdisk
The log tip vdisk (appropriate for certain environments, but not required for all) is a vdisk to which log records are initially written, then migrated to the log home vdisk. The intent is to use a small, high-performance NVRAM device for the log tip, and a larger vdisk on conventional spinning disks for the log home vdisk. The fast writes to the log tip hide the latency of the spinning disks used for the main body of the log.

log tip backup vdisk
The log tip backup vdisk (appropriate for certain environments, but not required for all) is used as an additional replica of the log tip vdisk when the log tip vdisk is two-way replicated on nonvolatile RAM disks. Ideally, the log tip backup vdisk provides a level of performance between that of NVRAM disks and that of spinning disks.

log reserved vdisk
Log reserved vdisks are optional vdisks that are used when the log home disk is not allocated in its own declustered array. Log reserved vdisks have the same size as the log home vdisk and are used to equalize the space consumption on the data declustered arrays, but they are otherwise unused.

Typical configurations

The following are descriptions of typical vdisk configurations in various recovery group environments:

ESS *without* NVRAM disks
In this configuration, a three-way replicated log tip vdisk is allocated on a declustered array made up of three SSDs. A four-way replicated log home vdisk is allocated in the first declustered array of HDDs.

ESS *with* NVRAM disks
In this configuration, a two-way replicated log tip vdisk is allocated on NVRAM disks, one from each of the servers.

A log tip backup vdisk is allocated on a declustered array of one or more SSDs. This provides an additional copy of the log tip data when one of the NVRAM disks is unavailable. If only one SSD is used, then the log tip backup uses a RAID code of `Unreplicated`.

A four-way replicated log home vdisk is allocated in the first declustered array of HDDs. A four-way replicated log reserved vdisk for each of the data declustered arrays that do not contain the log home vdisk.

ESS *with* NVRAM disks, using SSDs for data
In this configuration, a two-way replicated log tip vdisk is allocated on NVRAM disks, one from each of the servers.

All SSDs for a recovery group form a single declustered array, containing the log home vdisk and user data vdisks. No log tip backup disk is used.

Power 775 configuration
In this configuration, the log home vdisk is allocated on a declustered array made up of four SSDs. Only three-way and four-way replication codes are supported for the log home vdisk. In the typical system with four SSDs and with spare space equal to the size of one disk, the three-way replication code would be used for the log home vdisk

Creating vdisks and NSDs

You can create vdisks and NSDs using IBM Storage Scale RAID commands or using the ESS GUI.

After you create a vdisk using the **mmcrvdisk** command, you can create NSDs using the **mmcrnsd** command.

Alternatively, when you use the ESS GUI to create a file system, the GUI creates the vdisks and NSDs as well. NSDs and vdisks that are created using IBM Storage Scale RAID commands are ignored by the ESS GUI and cannot be used for creating a file system using the ESS GUI file system creation wizard, which is launched by using the **Create File System** action in the **Files > File Systems** view. Users who plan to create file systems using the ESS GUI must not create vdisks and NSDs using IBM Storage Scale RAID commands.

The relationship between vdisks and NSDs is as follows:

- GPFS file systems are built from vdisk NSDs in the same way as they are built from any other NSDs.
- While an NSD exists for a vdisk, that vdisk cannot be deleted.
- A node cannot serve vdisk-based NSDs and non-vdisk-based NSDs.
- Vdisk NSDs should not be used as tiebreaker disks.

For more information about:

- The **mmcrvdisk** command, see "mmcrvdisk command" on page 291
- The **mmcrnsd** command, see the *IBM Storage Scale: Command and Programming Reference*
- Vdisk and NSD best practices, see Appendix A, "Best-practice recommendations for IBM Storage Scale RAID," on page 259.

Vdisk states

IBM Storage Scale RAID reports vdisk states, which are displayed using the **mmlsvdisk** command and also on the **mmlsrecoverygroup** command if the `-L` option is used. You can also use the ESS GUI to display vdisk states. The state of vdisks is displayed in the **Arrays > Volumes** view.

Vdisks are normally in the ok state, which indicates that the vdisk is fully functional with full redundancy. When a pdisk failure affects a specific vdisk, the vdisk state will be reported as degraded until the affected tracks have been rebuilt onto spare space.

When enough pdisks have failed that the specific vdisk has no redundancy left, the vdisk state will be reported as `critical` until the critically affected tracks have been rebuilt onto spare space. If the system uses up all the available spare space while a vdisk is in the degraded or critical state, the state will be followed by `(need spare)`, which indicates that rebuild activity is stalled, and will resume once the failed pdisks have been replaced.

Table 6. Vdisk states

State	Description
ok	The vdisk is functioning normally.
m`/`*n* `-degraded`	The vdisk is currently running in degraded mode: ***m*** is the number of pdisks that are draining ***n*** is the fault-tolerance of the vdisk.
`critical`	The vdisk is currently running in degraded mode and can tolerate no more pdisk losses.
`(need spare)`	Rebuild will resume when more spare space is available; applies to degraded and critical states.

Determining pdisk-group fault-tolerance

You can obtain a synopsis of the current layout for user and system data by running this command:

```
mmlsrecoverygroup RecoveryGroupName -L
```

The output of this command includes a synopsis for the following:

- configuration data rebuild space
- configuration data recovery group descriptor layout
- configuration data system index layout
- user data vdisk layout

Note: The actual and maximum pdisk-group fault-tolerance values are point-in-time calculations based on the available disk space and current disk hardware configuration. These values could change whenever a rebuild or rebalance operation occurs.

Here is some sample output:

```
config data         declustered array   VCD spares     actual rebuild spare space         remarks
------------------  ------------------  -------------  ---------------------------------  ----------------
rebuild space       da0                 2              1 enclosure                        limited by VCD
spares

config data         max disk group fault tolerance     actual disk group fault tolerance  remarks
------------------  ---------------------------------  ---------------------------------  ----------------
rg descriptor       1 enclosure + 1 pdisk              1 enclosure + 1 pdisk
system index        2 enclosure                        1 enclosure                        limited by rebuild
space

vdisk               max disk group fault tolerance     actual disk group fault tolerance  remarks
------------------  ---------------------------------  ---------------------------------  ----------------
rg00log             2 enclosure                        1 enclosure                        limited by rebuild
space
rg00meta            3 enclosure                        1 enclosure                        limited by rebuild
space
rg00data            1 enclosure                        1 enclosure
```

This sample output from the `mmlsrecoverygroup` *RecoveryGroupName* `-L` command includes:

- a `config data` section with a `rebuild space` entry, which shows the available rebuild space that can be used to restore redundancy of the configuration data after disk failure
- a `config data` section with:
  - an `rg descriptor` entry, which shows the recovery group descriptor layout
  - a `system index` entry, which shows the system index layout
- a `vdisk` section, which shows a set of user-defined vdisk layouts.

This sample output lists the exact type of failure we would have survived in the `actual disk group fault tolerance` column:

```
vdisk         max disk group fault tolerance  actual disk group fault tolerance  remarks
----------  --------------------------------   --------------------------  --------------
.
.
.
rg00data       1 enclosure                       1 enclosure
```

For the `rg00data` vdisk, the output shows that we could survive one enclosure failure in the `actual disk group fault tolerance` column.

Note that for some vdisks in the sample output, there is a difference in maximum versus actual disk group fault tolerance:

```
vdisk       max disk group fault tolerance   actual disk group fault tolerance  remarks
----------  ------------------------------  --------------------------------  ----------------
.
.
.
rg00meta   3 enclosure                     1 enclosure                        limited by
rebuild space
```

This example indicates that even though the `rg00meta` vdisk has a layout that ensures its data is protected from three enclosure failures, as shown in the `max disk group fault tolerance` column, its actual disk group fault tolerance is one enclosure and it is being limited by rebuild space dependency. Effectively, the configuration data rebuild space is not sufficient and is limiting the disk group fault tolerance of its vdisk dependent.

In cases where the actual disk group fault tolerance is less than the maximum disk group fault tolerance, it is best to examine the dependency change that is shown in the `remarks` column. For this example, you would examine the `rebuild space` entry under the `config data` section:

```
config data    declustered array   VCD spares     actual rebuild spare space      remarks
-----------  ------------------  -------------  -----------------------------  ----------------
rebuild space    da0               2              1 enclosure                   limited by VCD
spares
```

This section indicates that we are being limited by the VCD spares. It is possible to increase the VCD spares and thus increase the actual rebuild spare space. If the rebuild space increases above 1 enclosure, it would no longer be the limiting dependency factor for the `rg00meta` vdisk. See “Increasing VCD spares” on page 61 for more information.

Note that the initial allocation during vdisk creation and subsequent redistribution of strips of the redundancy code in the event of failure or disk group changes is a dynamic operation. As such, it must work within the space available to it during the operation. The goals of this operation are always to minimize unnecessary movement, while at the same time balancing fault tolerance for all user and configuration data. Because of the vagaries of available space during initial allocation versus redistribution and because of the wide possible range of vdisk redundancy codes within the system, an initial fault tolerance state might not be repeatable after disk group faults and rebuilds occur.

For more information about **mmlsrecoverygroup** command output, see “mmlsrecoverygroup command” on page 327.

Chapter 5. Managing TRIM support for storage space reclamation

The TRIM feature ensures performance enhancement by enabling storage controllers to reclaim free space.

IBM Storage Scale RAID supports a TRIM operation to enhance the performance of flash-based declustered arrays. The TRIM feature allows the underlying storage controllers to reclaim free space from the file system. You can use the **mmreclaimspace** command to run TRIM commands on the storage media. The TRIM feature must be enabled at the physical disk level, within a declustered array, and at the file system NSD level.

Note: TRIM is only supported on NVMe-based NSDs.

Best practices for TRIM configurations

The suggested practices can be adopted for TRIM configuration.

The need for TRIM depends on the amount of data that is to be written and the availability of space that can be reclaimed at a time. A few ideal practices are suggested that can help you bring out the best benefits of the TRIM feature.

- It is advisable to run TRIM during a maintenance window or when the workload is light. For more information, see *mmreclaimspace command* in the *IBM Storage Scale: Command and Programming Reference*Guide.
- It is advisable to run the **mmreclaimspace** command with the **-P** option to define a specific storage pool. The **mmreclaimspace** command ignores the NSDs in the file system or the file system storage pool that are not configured for TRIM.
- TRIM can impact concurrent foreground workload bandwidth for write-heavy workloads, especially workloads that create many new files.
- The frequency of when to run TRIM depends on your workload. TRIM has the most benefit for workloads that routinely fill and empty the NSD/Vdisks on the ESS storage through file creation and deletion. TRIM has less benefit for workloads that mostly modify or update existing files.

Upgrading file system and recovery group from an earlier code level

Follow the procedure described to upgrade from an earlier code level.

Procedure

1. Follow the guidelines specific to your product and complete the upgrade process.
2. Issue the **mmchconfig release=LATEST** command.

   Note: The cluster level must be 5.0.5.1 or higher.
3. Issue the following command to update the recovery group to the latest version

   mmvdisk recoverygroup change --recovery-group rg --version LATEST
4. Issue the **mmlsfs** command with the **-V** option to ensure that the file system version is at least 5.0.4

   Note:

   - Issue the **mmchfs** command with the **-V** option to change the file system version.
   - For rolling upgrades, all nodes must be at the latest level or at a level that supports TRIM before it is enabled.

Enabling TRIM support on new recovery group and file system

Follow the procedure described to enable TRIM support while you create a new recovery group and file system.

Procedure

1. Run **mmvdisk recoverygroup create** command with the option **--trim-da nvme** to enable TRIM for a declustered array while you create a recovery group.
2. Run **mmvdisk filesystem create** command with the option **--trim auto** to enable TRIM for the new file system you are creating.

Enable, disable, extend and view TRIM settings on an existing declustered array and file system

You can enable, disable, extend or display TRIM support for an existing declustered array and file system.

To enable or disable TRIM on a IBM Storage Scale RAID system

TRIM can be dynamically enabled or disabled on an existing declustered array and file system. TRIM must be enabled on a declustered array before you enable it at the file system level. When you disable TRIM, you must first disable it from the file system NSDs, and then from the declustered array.

Issue the **mmvdisk recoverygroup change** command with the **--declustered-array --trim-da {yes|no}** to specify the required TRIM settings on the declustered array and recovery group.

Issue the **mmvdisk filesystem change** command with the **--trim {auto|no}** option to specify the required TRIM settings on the file system.

To extend a TRIM-enabled IBM Storage Scale RAID system

A pdisk automatically inherits the TRIM setting of the DA to which it is added. The TRIM setting can be inherited when you scale-up or scale-out the recovery groups. You can either add an IBM Storage Scale Erasure Code Edition (ECE) node by using the **mmvdisk recoverygroup add** command to scale-out the ECE recovery group, or you can scale-up the ECE or ESS recovery group by using the **mmvdisk recoverygroup resize** command.

For scale-out on ECE recovery group, the vdisk set will be extended automatically. The new vdisk NSDs in the vdisk set will inherit the TRIM setting in the file system.

For scale-up on ECE or ESS recovery group, issue the **mmvdisk filesystem add** command with the option **--trim {auto|no}** to select TRIM settings when you are adding a vdisk set to a file system.

Note: The default TRIM setting while adding a new vdisk set to a file system is "no" irrespective of the TRIM settings of the existing vdisk sets.

To View TRIM Settings

TRIM settings are enabled at the File system and DA level. View the TRIM settings with the following commands:

Issue the **mmvdisk recoverygroup list** command with the option **--da** to view the TRIM settings enabled for a declustered array. A sample is shown:

```
# mmvdisk recoverygroup list --recovery-group rg1 --da

declustered   needs               vdisks       pdisks          capacity
   array     service  type  trim  user log  total spare rt  total raw free raw  background task
-----------  -------  ----  ----  ---- ---  ----- ----- --  --------- --------  ---------------
DA1          no       NVMe  yes      0  13     12     2  2   3839 GiB 1190 GiB  scrub 14d (63%)
```

```
mmvdisk: Total capacity is the raw space before any vdisk set definitions.
mmvdisk: Free capacity is what remains for additional vdisk set definitions.
```

Issue the **mmvdisk filesystem list** command to view TRIM settings enabled for file system NSDs. A sample is shown:

```
# mmvdisk filesystem list --file-system mixfs

                                vdisk  with     list of       holds    holds  storage
vdisk set       recovery group  count  trim  failure groups  metadata  data    pool
--------------  --------------  -----  ----  --------------  --------  -----  -------
NVME            rg1                 4     4             1,2  no        yes    data
ESSDATA         rg1                 1     0               1  yes       yes    system
```

Note: In the preceding example, the NSDs from the "NVME" vdisk set have TRIM enabled. However, TRIM is not enabled in the vdisks of the "ESSDATA" vdiskset.

Reclaim space after enabling TRIM support

To reclaim space after you enable TRIM support, you must run the **mmreclaimspace** command.

About this task

Follow the procedure to manually run the **mmreclaimspace** command.

Procedure

1. Configure the file system and the recovery group.
2. Run the **mmreclaimspace** command to manually start the TRIM function. A sample is shown for running the command on an IBM Storage Scale Erasure Code Edition node.

```
[root@ece08]# mmreclaimspace  gpfs3 -P system --reclaim-threshold 80
disk                disk size  failure holds    holds           free in KB          free in
KB     reclaimed in KB
name                    in KB    group metadata data        in full blocks        in
fragments        in subblocks
--------------- ------------- -------- -------- ----- --------------------
------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 4.03 TB)
RG001LG001VS001     335984640        1 yes      yes       177795072 ( 53%)         19448
( 0%)      176283648 (52%)
RG001LG007VS001     335984640        1 yes      yes       177876992 ( 53%)         17336
( 0%)      176328704 (52%)
RG001LG003VS001     335984640        1 yes      yes       177967104 ( 53%)         15320
( 0%)      176422912 (53%)
RG001LG005VS001     335984640        1 yes      yes       177922048 ( 53%)         11352
( 0%)      176392192 (53%)
RG001LG004VS001     335984640        2 yes      yes       177922048 ( 53%)         17208
( 0%)      176369664 (52%)
RG001LG006VS001     335984640        2 yes      yes       177915904 ( 53%)         17304
( 0%)      176369664 (52%)
RG001LG002VS001     335984640        2 yes      yes       177790976 ( 53%)         15160
( 0%)      176257024 (52%)
RG001LG008VS001     335984640        2 yes      yes       177764352 ( 53%)         14936
( 0%)      176218112 (52%)
                -------------                           --------------------
------------------- -------------------
(pool total)       2687877120                              1422954496 ( 53%)        128064
( 0%)     1410641920 (52%)
                =============                           ====================
=================== ===================
(total)            2687877120                              1422954496 ( 53%)        128064
( 0%)     1410641920 (52%)
```

Note: The values of the **--reclaim-threshold** parameter can impact on performance. For more information about this parameter, see the *mmreclaimspace command* in the *IBM Storage Scale: Command and Programming Reference Guide* guide.

Automatic background TRIM

IBM Storage Scale supports automatic reclamation of free space when it is available after deletion.

This operation can be enabled by setting the **backgroundSpaceReclaimThreshold** parameter. The parameter specifies the percentage of reclaimable blocks that must occur in an allocation space for devices capable of space reclaim, such as NVMe and thin provisioned disks, to trigger a background space reclaim. The default value is 0 that indicates the background space reclamation is disabled. You can enable it by setting it to a value greater than 0 but less than or equal to 100. If a lower value is set, the free space is reclaimed frequently.

The lower value is disabled by default. Use the **mmchconfig** command to enable it. This parameter can be set at the cluster level. Therefore, it is recommended that the setting of **backgroundSpaceReclaimThreshold** only be applied on node-class or individual IBM Storage Scale node basis for the platforms in the following table. For more information about the **backgroundSpaceReclaimThreshold** parameter see, *mmchconfig command* in IBM Storage Scale: Command Reference.

This table does not contain IBM Storage Scale RAID platforms that do not support the NVMe-based storage. To enable TRIM, see Managing TRIM support for storage space reclamation.

Table 7. Background TRIM settings

IBM Storage Scale RAID platforms	The backgroundSpaceReclaimThreshold parameter value	Comments and limitations
ESS 3000	0[1]	Background TRIM is not supported.
ESS 3200	15	Background TRIM might degrade the performance of certain write workloads*.
ESS 3500 (performance/hybrid only)	15	Background TRIM might degrade the performance of certain write workloads*.

[1] This is a default value. When you want to disable the enabled background TRIM, set the value to 0 again.

* The space reclamation, whether background or manual, affects write workloads that allocate new blocks from the file system. For example, when you create and populate new files. If such workloads are the primary use case or occur at unscheduled times, then the automatic background file system TRIM might not be the best fit for such an environment. It is recommended to verify **backgroundSpaceReclaimThreshold** in your environment. Additionally, the setting can be disabled dynamically at any time. For more information about TRIM configuration, see Best practices for TRIM configurations.

Chapter 6. Managing the self-encrypting drives support on IBM Elastic Storage Server

The self-encrypting drives (SED) support protects data at rest on Elastic Storage System (ESS) drives. After this support is enabled, data on stolen or lost drives cannot be read without an authentication key, which is also called master encryption key (MEK).

- The SED support can be enabled only if all drives of a recovery group are SED capable.
- The SED support is available in Data Management Edition (DME) of IBM Storage Scale only.
- After the SED support enabled for a recovery group, you cannot disable it.
- If an MEK is lost, you lose access to entire data of a recovery group.

This section describes more details about setting an MEK, enabling the SED support, and changing an MEK.

Configuring a master encryption key

An MEK is used to enable SED support by enrolling recovery group drives to the SED support. With the same MEK, you can unlock the locked drives and crypto-erase the SED enabled drives. The ESS uses an external key manager such as GKLM for the MEK generation and key management. The **mmkeyserv** command can be used to set up an MEK for ESS I/O servers. For more details, see ***mmkeyserv command*** in *IBM Storage Scale: Command and Programming Reference Guide*.

Complete the following steps on an ESS I/O server node to configure an MEK:

1. Add a remote key management (RKM) server connection to ESS I/O server nodes.

```
# mmkeyserv server add
```

2. Create a tenant on the RKM server, and add the tenant to the ESS I/O server nodes.

```
# mmkeyserv tenant add
```

3. Creates a key client for ESS I/O server nodes to communicate with the RKM server.

```
# mmkeyserv client create
```

4. Register the key client to the tenant that is added to ESS I/O server nodes.

```
# mmkeyserv client register
```

5. Create encryption keys in the tenant.

```
# mmkeyserv key create
```

Enabling the SED support on a recovery group

Complete the following tasks to enable the SED support on a specified recovery group:

1. Before you enroll a recovery group for the SED support, verify whether all the drives except NVRAM drives of the recovery group are SED capable or not.

```
# mmvdisk sed verify  {--all | --recovery-group RgName[,RgName...] |
                      --recovery-group RgName [--pdisk pdiskname] |
                      --pdisk-path pdisk-path}
```

--all
Selects all recovery groups of the cluster.

--recovery-group
Specifies the recovery group name(s).

--pdisk
Specifies the pdisk path.

2. Create an MEK.

```
# mmkeyserv key create
```

3. Get the MEK key Id.

```
# mmkeyserv key show
```

4. Enable the SED support on a specified recovery group.

```
# mmvdisk sed enroll --recovery-group RecoveryGroupName --rkmid RKMId --key-uuid KeyId
```

--rkmid
Specifies a new RKM ID that is displayed as a part of the **mmkeyserv rkm show** command.

--key-uuid
Specifies the MEK UUID.

--recovery-group
Specifies the name of a recovery group name whose drives would be enabled for SED support.

The command can run longer depending on the number of drives in the recovery group.

5. If the **mmvdisk** command is interrupted, rerun the command. When the command is rerun, enrolled drives are skipped and drives that were not enrolled before are enrolled with the same MEK.
6. When the SED support is enabled on all drives, a new `sedKeyId` configuration key is set with the MEK Key ID and RKM key ID to enable the SED support. The `sedKeyId` value is concatenation of MEK Key ID, : (colon) and RKM Key ID. The `sedKeyId` configuration keyword is set for the node class of a recovery group only.
7. Check whether all the drives of a recovery group are enabled for the SED support.

```
#mmvdisk sed list {--all | --recovery-group RgName[,RgName...] |
                   --recovery-group RgName [--pdisk pdiskname] |
                   --pdisk-path pdisk-path}
```

--all
Selects all recovery groups of the cluster.

--recovery-group
Specifies the recovery group name(s).

--pdisk
Specifies the pdisk path.

This command displays whether the SED support is enabled by using an MEK specified by **sedKeyId** or some other unknown key. It also displays whether the drive is locked or unlocked.

Changing the MEK for recovery group drives

After the SED support is enabled for a recovery group, you might have to change the MEK for all the drives of the recovery group. The MEK needs to be changed if the current MEK is compromised or some organization policies such as MEK needs to be changed periodically. The MEK SED can be changed at the recovery group level on the ESS systems only.

1. Set up a new MEK before you change the MEK for a recovery group.
2. Get a new MEK key ID.

```
# mmkeyserv key show
```

3. Get an RKM ID.

```
# mmkeyserv rkm show
```

4. Change the MEK of the specified recovery group.

```
# mmvdisk sed rekey --recovery-group RecoveryGroupName --rkmid RKMId --key-uuid KeyId
```

--rkmid
Specifies a new RKM ID displayed as part of the **mmkeyserv rkm show** command.

--key-uuid
Specifies the MEK UUID.

--recovery-group
Specifies the recovery group name whose drives would be enabled for the SED support.

This command might run longer depending on the number of drives in the recovery group.

5. If the **mmvdisk** command is interrupted, rerun to restart the rekey the MEK of the recovery group.

This command changes MEK for the drives that were not rekeyed before with the same new key. After all the drives are rekeyed successfully, the sedKeyId configuration key is updated with a new MEK Key Id and RKM key Id values (concatenation of new MEK Key ID, : and new RKM Key ID).

Migrating a recovery group for the SED support

The older existing recovery groups, which already have some user data, can be enabled for the SED support by using the migration process. The migration can be done on a live system when a workload is going on.

To enable SED support on all drives of a recovery group, complete the following steps:

1. Configure an MEK and find the MEK Key ID (KeyId) and RKM ID.

```
# mmkeyserv
```

For more details, see ***mmkeyserv command*** in *IBM Storage Scale: Command and Programming Reference Guide*.

2. Ensure that all the drives of the recovery group are in the OK state.
3. Verify that all drives of the recovery group are SED capable.

```
# mmvdisk sed verify
```

4. Run the **mmvdisk sed enroll** (with a hyper link) command that passes the MEK key ID (KeyId) and RKM ID (RKMId) that found in step 1 as shown in the following command:

```
# mmvdisk sed enroll  --recovery-group RgName --rkmid RKMid --key-uuid KeyId
```

- The new <KeyId:RKMid> value is stored in the sedKeyId config variable of node class.
- The MEK key is updated from default MSID to the new key specified on all drives, and the SED support is enabled on all drives.

5. If the enroll (migration) process is stopped or interrupted, rerun the **sed enroll** command to continue the migration.

Auto unlock of the recovery group drives

When the SED support is enabled, the SED capable drives are locked after these are power-recycled. The drives on a recovery group are unlocked by using the following methods:

- The current MEK that is set by using the SedKeyId keyword is used to unlock drives.
- The disk hospital uses the current MEK when it detects SED locked drives.
- During the daemon startup, drives are unlocked by using the current MEK with the **mmstartup** command.

Auto enroll new recovery group drives

After the SED support is enabled on a recovery group, the new drives that are added as a part of disk replacement procedure are automatically enrolled. The current MEK that is specified by using the `sedKeyId` configuration keyword used to enroll drives.

Crypto-erase of recovery group drives

After the SED support is enabled on a recovery group, crypto-erase is used on the SED capable drives of the recovery group when deleting the recovery group or individual drives. The crypto-erase is a mandatory operation during the recovery group deletion and disk deletion from the recovery group.

Note: After the crypto-erase of drives, you cannot get the old data back from drives. To reuse the drive again, the drives must be formatted again.

Chapter 7. Configuring components on the Elastic Storage Server

In an ESS system, IBM Storage Scale uses component information in the cluster configuration data to perform configuration and validation tasks. For example, when a disk needs to be replaced, the system reports the rack location of the enclosure containing the disk along with a description of the disk's slot within the enclosure.

A component in this context refers to some resource, usually hardware, that is part of the GPFS cluster; for example, `rack` or `storage enclosure`. Each component has an associated component specification that is identified by its part number. You can use the `mmlscompspec` command to see the defined component specifications.

You will normally configure components when deploying a cluster or when adding new hardware to a cluster. The cluster must exist before you can run any of the component-related commands.

If you use IBM Storage Scale RAID commands to configure components, the basic configuration steps are as follows:

1. Add components to the configuration using `mmdiscovercomp` and `mmaddcomp`.
2. Define the location of storage enclosures and servers using `mmchcomploc`.
3. Set the two-digit ID visible on the back of some storage enclosures using `mmsyncdisplayid`.
4. Update the component names and other attributes using `mmchcomp`.

The following commands use or change the cluster's component configuration:

mmaddcomp
Adds new components

mmchcomp
Change component attributes

mmchcomploc
Change component locations

mmdelcomp
Delete components

mmdiscovercomp
Discover components and add them to the configuration

mmlscomp
List components.

mmlscomploc
List component locations

mmlscompspec
List component specifications

mmsyncdisplayid
Synchronize enclosure display IDs with the cluster configuration

For information about the options that these commands support, see Appendix B, “IBM Storage Scale RAID commands,” on page 263. Most commands allow you to make a single change using command-line arguments. They also support stanza file input for making any number of changes with a single command.

Alternatively, instead of using the IBM Storage Scale RAID commands described in this chapter, you can optionally perform initial component configuration using the ESS GUI's system setup wizard, which is launched automatically after you log in to the ESS GUI for the first time.

You can use the ESS GUI to edit component configuration at a later time. You can do this in the **Actions > Edit Rack Components** view or the **Monitoring > System** view.

Adding components to the cluster's configuration

This section discusses how to add components to the cluster's configuration in an ESS system. Sample output in this topic might not match the output produced on your system.

Before adding any components, you might want to view the available component specifications. To do so, you can use the **mmlscompspec** command, which lists the defined component specifications, identified by part number.

```
$ mmlscompspec

    Rack Specifications

Part Number  Height  Description
-----------  ------  -------------------------------------------------
1410HEA          42  42U 1200mm Deep Expansion Rack
1410HPA          42  42U 1200mm Deep Primary Rack

    Server Specifications

Part Number  Height  Description
-----------  ------  -------------------
824722L           2 IBM Power System S822L

    Storage Enclosure Specifications

Part Number  Height  Description                  Vendor ID  Product ID  Drive Slots  Has
Display ID
-----------  ------  ---------------------------  ---------  ----------  -----------
--------------
1818-80E          4  DCS3700 Expansion Enclosure  IBM        DCS3700
60               1
```

If this was the first component command run on the cluster, you will see some initialization messages before the command output. The list of component specifications will be short because it only includes supported ESS hardware.

You can use **mmdiscovercomp** to define some of the cluster's components. This command finds supported storage enclosures attached to any of the cluster nodes and adds them to the cluster's configuration. (In your version of IBM Storage Scale, **mmdiscovercomp** might find additional component types.) The output below resembles a cluster that includes two GL4 units.

```
$ mmdiscovercomp all
Adding enclosure: serial number = SV12345001; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345007; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345003; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345005; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345004; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345002; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345008; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345006; vendor ID = IBM; product ID = DCS3700

Storage Enclosures

Status  Comp ID  Component Type    Part Number  Serial Number  Name                Display ID
------  -------  ----------------  -----------  -------------  ------------------  ----------
new           1  storageEnclosure  1818-80E     SV12345001     1818-80E-SV12345001  00
new           2  storageEnclosure  1818-80E     SV12345007     1818-80E-SV12345007  00
new           3  storageEnclosure  1818-80E     SV12345003     1818-80E-SV12345003  00
new           4  storageEnclosure  1818-80E     SV12345005     1818-80E-SV12345005  00
new           5  storageEnclosure  1818-80E     SV12345004     1818-80E-SV12345004  00
new           6  storageEnclosure  1818-80E     SV12345002     1818-80E-SV12345002  00
new           7  storageEnclosure  1818-80E     SV12345008     1818-80E-SV12345008  00
new           8  storageEnclosure  1818-80E     SV12345006     1818-80E-SV12345006  00
```

In this example, **mmdiscovercomp** found eight storage enclosures that were not already in the cluster configuration. Notice that each component gets a default name. Below we will discuss how you can change the name to some identifier that is more suitable for your environment. You may run **mmdiscovercomp** a second time, in which case it should not find any new components.

Note: The serial numbers used by IBM Storage Scale to identify 1818-80E storage enclosures do not match the product serial numbers shown on the outside of the enclosure. You may want to record this

visible serial number as part of the enclosure's name. For example, you could rename an enclosure to S1E3-SX98760044, which would stand for "storage server 1" (S1), "enclosure 3" (E3), with product serial number SX98760044. For details about how to change the name of a component using the **mmchcomp** command, see “Updating component attributes” on page 78.

The other essential components are racks to hold the storage enclosures. Use the **mmlscompspec** command to see the available rack part numbers, then use the **mmaddcomp** command to define the racks. This example shows how to define a "42U 1200mm Deep Primary Rack" and give it the name R01C01.

```
$ mmaddcomp 1410HPA --name R01C01
$ mmlscomp

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HPA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  1818-80E     SV12345001     1818-80E-SV12345001
      2  1818-80E     SV12345007     1818-80E-SV12345007
      3  1818-80E     SV12345003     1818-80E-SV12345003
      4  1818-80E     SV12345005     1818-80E-SV12345005
      5  1818-80E     SV12345004     1818-80E-SV12345004
      6  1818-80E     SV12345002     1818-80E-SV12345002
      7  1818-80E     SV12345008     1818-80E-SV12345008
      8  1818-80E     SV12345006     1818-80E-SV12345006
```

Defining component locations

This section discusses how to define component locations in an ESS system. Sample output in this topic might not match the output produced on your system.

Use the `mmchcomploc` command to place a component in a rack or change its location. This example puts the storage enclosure SV12345003 in rack R01C01 at U location 13.

```
$ mmchcomploc SV12345003 R01C01 13
$ mmlscomploc
Component            Location
-------------------  -------------------
1818-80E-SV12345003  Rack R01C01 U13-16
```

The first argument, which identifies the component, can be either the component ID, serial number, or name. The second argument is the rack (container) and the third argument is the U position in the rack.

Changing one component at a time can be slow because each change is propagated to all nodes in the cluster. You may want to use a stanza file to make multiple changes with a single command. The following example uses a stanza file to position the remaining enclosures:

```
$ cat comploc.stanza
%compLoc: compId=SV12345007 containerId=R01C01 position=1
%compLoc: compId=SV12345005 containerId=R01C01 position=5
%compLoc: compId=SV12345003 containerId=R01C01 position=13
%compLoc: compId=SV12345008 containerId=R01C01 position=17
%compLoc: compId=SV12345001 containerId=R01C01 position=21
%compLoc: compId=SV12345002 containerId=R01C01 position=25
%compLoc: compId=SV12345006 containerId=R01C01 position=33
%compLoc: compId=SV12345004 containerId=R01C01 position=37

$ mmchcomploc -F comploc.stanza
$ mmlscomploc
Component            Location
-------------------  -------------------
1818-80E-SV12345007  Rack R01C01 U01-04
1818-80E-SV12345005  Rack R01C01 U05-08
1818-80E-SV12345003  Rack R01C01 U13-16
1818-80E-SV12345008  Rack R01C01 U17-20
1818-80E-SV12345001  Rack R01C01 U21-24
1818-80E-SV12345002  Rack R01C01 U25-28
```

```
1818-80E-SV12345006  Rack R01C01 U33-36
1818-80E-SV12345004  Rack R01C01 U37-40
```

Synchronizing display IDs

This section discusses how to synchronize display IDs in an ESS system. Sample output in this topic might not match the output produced on your system.

Some ESS disk enclosures have a two-digit display on their environmental service modules (ESMs). This display is visible from the back of the enclosure. When configured correctly, the two-digit display will be the same as the U location of the enclosure within its rack. `mmsyncdisplayid` sets the display on the ESMs to match the locations of the enclosure as defined by the component configuration. Here is an example:

```
$ mmsyncdisplayid all
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number   Serial Number  Name                 Display ID
-------  -----------   -------------  -------------------  ----------
      1  1818-80E      SV12345001     1818-80E-SV12345001  21
      2  1818-80E      SV12345007     1818-80E-SV12345007  01
      3  1818-80E      SV12345003     1818-80E-SV12345003  13
      4  1818-80E      SV12345005     1818-80E-SV12345005  05
      5  1818-80E      SV12345004     1818-80E-SV12345004  37
      6  1818-80E      SV12345002     1818-80E-SV12345002  25
      7  1818-80E      SV12345008     1818-80E-SV12345008  17
      8  1818-80E      SV12345006     1818-80E-SV12345006  33
```

With the display ID set on the actual hardware, you should now verify that the locations defined in the cluster's configuration actually match the enclosure's position in its rack. If you find that the enclosures are not numbered 01, 05, 13, ... starting from the bottom of the rack, then use the `mmchcomploc` command to correct the configuration. Then rerun `mmsyncdisplayid` and recheck the hardware.

Updating component attributes

This section discusses how to update component attributes in an ESS system. Sample output in this topic might not match the output produced on your system.

You can change some component attributes such as the name or serial number. This example updates the name of the third enclosure so that it includes the product serial number as recommended:

```
$ mmchcomp 3 --name S1E3-SX98760044
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number  Serial Number   Name                 Display ID
-------  -----------  -------------   -------------------  ----------
      1  1818-80E      SV12345001     1818-80E-SV12345001  21
      2  1818-80E      SV12345007     1818-80E-SV12345007  01
      3  1818-80E      SV12345003     S1E3-SX98760044      13
      4  1818-80E      SV12345005     1818-80E-SV12345005  05
      5  1818-80E      SV12345004     1818-80E-SV12345004  37
      6  1818-80E      SV12345002     1818-80E-SV12345002  25
      7  1818-80E      SV12345008     1818-80E-SV12345008  17
      8  1818-80E      SV12345006     1818-80E-SV12345006  33
```

The `mmchcomp` *Component* argument may be either the component ID, the component serial number, or the component name. This example uses the component ID (the value 3).

Changing one component at a time can be slow, so you can use a stanza file to make multiple changes with a single command. As an aid, `mmlscomp` can list components in stanza format. You can capture this

output into a file, edit the file to make the desired updates, then use the stanza file as input to `mmchcomp`. The following example uses this procedure to rename the remaining enclosures:

```
$ mmlscomp --format stanza > rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='1818-80E-SV12345001'
  partNumber=1818-80E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='1818-80E-SV12345007'
  partNumber=1818-80E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='S1E3-SX98760044'
  partNumber=1818-80E
  serialNumber=SV12345003

...
```

Edit `rename.stanza` to define new enclosure names, then apply the changes using `mmchcomp`.

```
$ vi rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  name=R01C01
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='S1E5-SX98760048'
  partNumber=1818-80E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='S1E1-SX98760044'
  partNumber=1818-80E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='S1E3-SX98760041'
  partNumber=1818-80E
  serialNumber=SV12345003

...

$ mmchcomp -F comp.stanza
$ mmlscomp --sort name

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
```

```
      9  1410HEA                    R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number   Name                Display ID
-------  -----------  -------------   ------------------  ----------
      2  1818-80E      SV12345007     S1E1-SX98760044     01
      4  1818-80E      SV12345005     S1E2-SX98760049     05
      3  1818-80E      SV12345003     S1E3-SX98760041     13
      7  1818-80E      SV12345008     S1E4-SX98760036     17
      1  1818-80E      SV12345001     S1E5-SX98760048     21
      6  1818-80E      SV12345002     S1E6-SX98760050     25
      8  1818-80E      SV12345006     S1E7-SX98760039     33
      5  1818-80E      SV12345004     S1E8-SX98760052     37
```

Chapter 8. Monitoring IBM Storage Scale RAID

Monitoring the ESS system includes system health, performance, and capacity monitoring. You can monitor the system either through ESS GUI or with the help of CLI.

The following table lists the CLI options that are available to monitor the system.

Table 8. CLI options that are available to monitor the system. CLI options that are available to monitor the system	
CLI commands	**Function**
mmhealth	Single command that can monitor the health of nodes, services, logical components, events, list thresholds, and so on.
mmperfmon	Configures the performance monitoring tool and lists the performance metrics.
mmpmon	Manages performance monitoring of GPFS and displays performance information.
mmlsrecoverygroup	Lists information about IBM Storage Scale RAID recovery groups.
mmlspdisk	Lists information for one or more IBM Storage Scale RAID pdisks.
mmlsvdisk	Lists information for one or more IBM Storage Scale RAID vdisks.
mmlsenclosure	Displays the environmental status of IBM Storage Scale RAID disk enclosures.
mmaddcallback	Registers a user-defined command that GPFS will execute when certain events occur.

The following topics describe the monitoring options that are available in the ESS GUI:

- “Monitoring system health by using ESS GUI” on page 88
- “Performance monitoring using ESS GUI” on page 94
- “Monitoring capacity through GUI” on page 108

System health monitoring

The monitoring framework that is designed in the system takes care of the system health monitoring and you can use the **mmhealth** command to get the details collected by the monitoring framework.

Use the **mmhealth** command to get details of the following aspects:

- Health of individual nodes and entire set of nodes at the cluster level
- Health of the logical components
- Health of services that are hosted in the system
- Events that are reported in the system
- Thresholds that are defined in the system

For more information about the **mmhealth** command, see *IBM Storage Scale: Command and Programming Reference Guide*.

Hardware components are monitored through xtreme Cluster/Cloud Administration Toolkit (xCAT). It is a scalable and open-source cluster management software. The management infrastructure of ESS is deployed by xCAT.

Monitoring framework

Every service that is hosted on an ESS node has its own health monitoring service. The monitoring service works based on the roles that are assigned to the nodes in the cluster. The role of a node in monitoring determines the components that need to be monitored. The node roles are hardcoded. It also determines the monitoring service that is required on a specific node. For example, you do not need a CES monitoring on a non-CES node. The monitoring services are only started if a specific node role is assigned to the node. Every monitoring service includes at least one monitor.

The following list provides the details of the monitoring services available in the IBM Storage Scale system:

GPFS
Node role: Active on all ESS nodes.

Tasks: Monitors all GPFS daemon-related functions. For example, **mmfsd** process and *gpfs port accessibility*.

Network
Node role: Every ESS node has the node role that is required for the network monitoring service.

CES
Node role: Nodes with node class *cesNodeperforms* the monitoring services for all CES services. The CES service does not have its own monitoring service or events. The status of the CES is an aggregation of the status of the subservices.

File authentication
Tasks: Monitors LDAP, AD, or NIS-based authentication services.

Object authentication
Tasks: Monitors the OpenStack identity service functions.

Block
Tasks: Checks whether the iSCSI daemon is functioning properly.

CES network
Tasks: Monitors CES network-related adapters and IP addresses.

NFS
Tasks: Monitoring NFS-related functions.

Object
Tasks: Monitors the ESS for object functions. Especially, the status of relevant system services and accessibility to ports are checked.

SMB
Tasks: Monitoring SMB-related functions like the smbd process, the ports, and ctdb processes.

Cloud gateway
Node role: A node gets the cloud gateway node role if it is identified as a transparent cloud tiering node. These nodes are listed when you issue the **mmcloudgateway nodelist** command.

Tasks: Checks whether the cloud gateway service functions as expected.

Disk
Node role: Nodes with node class *nsdNodes* monitor the disk service.

Tasks: Checks whether the ESS disks are available and running.

File system
Node role: This node role is active on all ESS nodes.

Tasks: Monitors different aspects of IBM Storage Scale file systems.

GUI
Node role: Nodes with node class *GUI_MGMT_SERVERS* monitor the GUI service.

Tasks: Verifies whether the GUI services are functioning properly.

Hadoop connector
Node role: Nodes where the Hadoop service is configured get the Hadoop connector node role.

Tasks: Monitors the Hadoop data node and name node services.

Performance monitoring
Node role: Nodes where *PerfmonSensors* or *PerfmonCollectorservices* are installed get the performance monitoring node role. Performance monitoring sensors are determined through the *perfmon* designation in the **mmlscluster**. Performance monitoring collectors are determined through the *colCandidates* line in the configuration file.

Tasks: Monitors whether the performance monitoring collectors and sensors are running as expected.

For more information on system health monitoring options that are available in GUI, see "Monitoring system health by using ESS GUI" on page 88.

Monitoring events

The recorded events are stored in local database on each node. The user can get a list of recorded events by using the **mmhealth node eventlog** command. You can also use the **mmces** command to get the details of the events that are recoded in the system.

For more information about the **mmhealth** and **mmces** commands, see the *IBM Storage Scale: Command and Programming Reference*.

Monitoring events through callhome

The call home feature automatically notifies IBM Support if certain types of events occur in the system. Using this information, IBM Support can contact the system administrator in case of any issues. Configuring call home reduces the response time for IBM Support to address the issues.

The details are collected from individual nodes that are marked as call home child nodes in the cluster. The details from each child node are collected by the call home node. You need to create a call home group by grouping call home child nodes. One of the nodes in the group is configured as the call home node and it performs data collection and upload.

ESA call home is triggered for the following events only:

- power_supply_failed
- fan_failed
- can_fan_failed
- bootdrive_smart_failed
- bootdrive_missing
- canister_failed
- bootdrive_mirror_failed
- bootdrive_mirror_degraded
- can_temp_sensor_failed
- fan_speed_high
- fan_speed_low
- power_high_current
- power_high_voltage
- temp_bus_failed
- temp_sensor_failed
- voltage_bus_failed

- voltage_sensor_failed
- can_temp_bus_failed
- coin_battery_low
- coin_battery_missing
- cpu_unit_missing
- cpu_unit_speed_wrong
- dimm_module_missing
- dimm_module_size_wrong
- dimm_module_speed_wrong
- pair_canister_failed

The data gathering and upload can be configured individually on each group. Use the groups to reflect logical units in the cluster. For example, it is easier to manage when you create a group for all CES nodes and another group for all non-CES nodes.

Use the **mmcallhome** command to configure the call home feature in the system. For more information about the **mmcallhome** command, see the *IBM Storage Scale: Command and Programming Reference*.

IBM Storage Scale RAID callbacks

IBM Storage Scale RAID includes callbacks for events that can occur during recovery group operations. These callbacks can be installed by the system administrator using the `mmaddcallback` command.

The callbacks are provided primarily as a method for system administrators to take notice when important IBM Storage Scale RAID events occur. For example, an IBM Storage Scale RAID administrator can use the `pdReplacePdisk` callback to send an e-mail to notify system operators that the replacement threshold for a declustered array was reached and that pdisks must be replaced. Similarly, the `preRGTakeover` callback can be used to inform system administrators of a possible server failover.

As notification methods, no real processing should occur in the callback scripts. IBM Storage Scale RAID callbacks should not be installed for synchronous execution; the default of asynchronous callback execution should be used in all cases. Synchronous or complicated processing within a callback might delay GPFS daemon execution pathways and cause unexpected and undesired results, including loss of file system availability.

The IBM Storage Scale RAID callbacks and their corresponding parameters follow:

preRGTakeover
The `preRGTakeover` callback is invoked on a recovery group server prior to attempting to open and serve recovery groups. The `rgName` parameter may be passed into the callback as the keyword value `_ALL_`, indicating that the recovery group server is about to open multiple recovery groups; this is typically at server startup, and the parameter `rgCount` will be equal to the number of recovery groups being processed. Additionally, the callback will be invoked with the `rgName` of each individual recovery group and an `rgCount` of 1 whenever the server checks to determine whether it should open and serve recovery group `rgName`.

The following parameters are available to this callback: `%myNode`, `%rgName`, `%rgErr`, `%rgCount`, and `%rgReason`.

postRGTakeover
The `postRGTakeover` callback is invoked on a recovery group server after it has checked, attempted, or begun to serve a recovery group. If multiple recovery groups have been taken over, the callback will be invoked with `rgName` keyword `_ALL_` and an `rgCount` equal to the total number of involved recovery groups. The callback will also be triggered for each individual recovery group.

The following parameters are available to this callback: `%myNode`, `%rgName`, `%rgErr`, `%rgCount`, and `%rgReason`.

preRGRelinquish
The preRGRelinquish callback is invoked on a recovery group server prior to relinquishing service of recovery groups. The rgName parameter may be passed into the callback as the keyword value _ALL_, indicating that the recovery group server is about to relinquish service for all recovery groups it is serving; the rgCount parameter will be equal to the number of recovery groups being relinquished. Additionally, the callback will be invoked with the rgName of each individual recovery group and an rgCount of 1 whenever the server relinquishes serving recovery group rgName.

The following parameters are available to this callback: %myNode, %rgName, %rgErr, %rgCount, and %rgReason.

postRGRelinquish
The postRGRelinquish callback is invoked on a recovery group server after it has relinquished serving recovery groups. If multiple recovery groups have been relinquished, the callback will be invoked with rgName keyword _ALL_ and an rgCount equal to the total number of involved recovery groups. The callback will also be triggered for each individual recovery group.

The following parameters are available to this callback: %myNode, %rgName, %rgErr, %rgCount, and %rgReason.

rgOpenFailed
The rgOpenFailed callback will be invoked on a recovery group server when it fails to open a recovery group that it is attempting to serve. This may be due to loss of connectivity to some or all of the disks in the recovery group; the rgReason string will indicate why the recovery group could not be opened.

The following parameters are available to this callback: %myNode, %rgName, %rgErr, and %rgReason.

rgPanic
The rgPanic callback will be invoked on a recovery group server when it is no longer able to continue serving a recovery group. This may be due to loss of connectivity to some or all of the disks in the recovery group; the rgReason string will indicate why the recovery group can no longer be served.

The following parameters are available to this callback: %myNode, %rgName, %rgErr, and %rgReason.

pdFailed
The pdFailed callback is generated whenever a pdisk in a recovery group is marked as dead, missing, failed, or readonly.

The following parameters are available to this callback: %myNode, %rgName, %daName, %pdName, %pdLocation, %pdFru, %pdWwn, and %pdState.

pdRecovered
The pdRecovered callback is generated whenever a missing pdisk is rediscovered.

The following parameters are available to this callback: %myNode, %rgName, %daName, %pdName, %pdLocation, %pdFru, and %pdWwn.

pdReplacePdisk
The pdReplacePdisk callback is generated whenever a pdisk is marked for replacement according to the replace threshold setting of the declustered array in which it resides.

The following parameters are available to this callback: %myNode, %rgName, %daName, %pdName, %pdLocation, %pdFru, %pdWwn, %pdState, and %pdPriority.

pdPathDown
The pdPathDown callback is generated whenever one of the block device paths to a pdisk disappears or becomes inoperative. The occurrence of this event can indicate connectivity problems with the JBOD array in which the pdisk resides.

The following parameters are available to this callback: %myNode, %rgName, %daName, %pdName, %pdPath, %pdLocation, %pdFru, and %pdWwn.

daRebuildFailed
The daRebuildFailed callback is generated when the spare space in a declustered array has been exhausted, and vdisk tracks involving damaged pdisks can no longer be rebuilt. The occurrence of this event indicates that fault tolerance in the declustered array has become degraded and that disk maintenance should be performed immediately. The daRemainingRedundancy parameter indicates how much fault tolerance remains in the declustered array.

The following parameters are available to this callback: %myNode, %rgName, %daName, and %daRemainingRedundancy.

nsdCksumMismatch
The nsdCksumMismatch callback is generated whenever transmission of vdisk data by the NSD network layer fails to verify the data checksum. This can indicate problems in the network between the GPFS client node and a recovery group server. The first error between a given client and server generates the callback; subsequent callbacks are generated for each ckReportingInterval occurrence.

The following parameters are available to this callback: %myNode, %ckRole, %ckOtherNode, %ckNSD, %ckReason, %ckStartSector, %ckDataLen, %ckErrorCountClient, %ckErrorCountNSD, and %ckReportingInterval.

IBM Storage Scale RAID recognizes the following variables (in addition to the %myNode variable, which specifies the node where the callback script is invoked):

%ckDataLen
The length of data involved in a checksum mismatch.

%ckErrorCountClient
The cumulative number of errors for the client side in a checksum mismatch.

%ckErrorCountServer
The cumulative number of errors for the server side in a checksum mismatch.

%ckErrorCountNSD
The cumulative number of errors for the NSD side in a checksum mismatch.

%ckNSD
The NSD involved.

%ckOtherNode
The IP address of the other node in an NSD checksum event.

%ckReason
The reason string indicating why a checksum mismatch callback was invoked.

%ckReportingInterval
The error-reporting interval in effect at the time of a checksum mismatch.

%ckRole
The role (client or server) of a GPFS node.

%ckStartSector
The starting sector of a checksum mismatch.

%daName
The name of the declustered array involved.

%daRemainingRedundancy
The remaining fault tolerance in a declustered array.

%pdFru
The FRU (field replaceable unit) number of the pdisk.

%pdLocation
The physical location code of a pdisk.

%pdName
The name of the pdisk involved.

%pdPath
The block device path of the pdisk.

%pdPriority
The replacement priority of the pdisk.

%pdState
The state of the pdisk involved.

%pdWwn
The worldwide name of the pdisk.

%rgCount
The number of recovery groups involved.

%rgErr
A code from a recovery group, where 0 indicates no error.

%rgName
The name of the recovery group involved.

%rgReason
The reason string indicating why a recovery group callback was invoked.

All IBM Storage Scale RAID callbacks are local, which means that the event triggering the callback occurs only on the involved node or nodes, in the case of `nsdCksumMismatch`, rather than on every node in the GPFS cluster. The nodes where IBM Storage Scale RAID callbacks should be installed are, by definition, the recovery group server nodes. An exception is the case of `nsdCksumMismatch`, where it makes sense to install the callback on GPFS client nodes as well as recovery group servers.

A sample callback script, `/usr/lpp/mmfs/samples/vdisk/gnrcallback.sh`, is available to demonstrate how callbacks can be used to log events or email an administrator when IBM Storage Scale RAID events occur.

Logged events would look something like:

```
Fri Feb 28 10:22:17 EST 2014:  mmfsd: [W] event=pdFailed node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL daName=DA1 pdName=e4d5s03 pdLocation=SV13306129-5-3 pdFru=46W6911
pdWwn=naa.5000C50055D4D437 pdState=dead/systemDrain
Fri Feb 28 10:22:39 EST 2014:  mmfsd: [I] event=pdRecovered node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL daName=DA1 pdName=e4d5s03 pdLocation=SV13306129-5-3 pdFru=46W6911
pdWwn=naa.5000C50055D4D437 pdState=UNDEFINED
Fri Feb 28 10:23:59 EST 2014:  mmfsd: [E] event=rgPanic node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL rgErr=756 rgReason=missing_pdisk_causes_unavailability
Fri Feb 28 10:24:00 EST 2014:  mmfsd: [I] event=postRGRelinquish node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL rgErr=0 rgReason=unable_to_continue_serving
Fri Feb 28 10:24:00 EST 2014:  mmfsd: [I] event=postRGRelinquish node=c45f01n01-ib0.gpfs.net
rgName=_ALL_ rgErr=0 rgReason=unable_to_continue_serving
Fri Feb 28 10:35:06 EST 2014: mmfsd: [I] event=postRGTakeover node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL rgErr=0 rgReason=retry_takeover
Fri Feb 28 10:35:06 EST 2014: mmfsd: [I] event=postRGTakeover node=c45f01n01-ib0.gpfs.net
rgName=_ALL_ rgErr=0 rgReason=none
```

An email notification would look something like this:

```
> mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/root": 7 messages 7 new
>N  1 root                  Fri Feb 28 10:22  18/817   "[W] pdFailed"
 N  2 root                  Fri Feb 28 10:22  18/816   "[I] pdRecovered"
 N  3 root                  Fri Feb 28 10:23  18/752   "[E] rgPanic"
 N  4 root                  Fri Feb 28 10:24  18/759   "[I] postRGRelinquish"
 N  5 root                  Fri Feb 28 10:24  18/758   "[I] postRGRelinquish"
 N  6 root                  Fri Feb 28 10:35  18/743   "[I] postRGTakeover"
 N  7 root                  Fri Feb 28 10:35  18/732   "[I} postRGTakeover"

From root@c45f01n01.localdomain  Wed Mar  5 12:27:04 2014
Return-Path: <root@c45f01n01.localdomain>
X-Original-To: root
Delivered-To: root@c45f01n01.localdomain
Date: Wed, 05 Mar 2014 12:27:04 -0500
To: root@c45f01n01.localdomain
Subject: [W] pdFailed
```

```
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
From: root@c45f01n01.localdomain (root)
Status: R

Wed Mar 5 12:27:04 EST 2014: mmfsd: [W] event=pdFailed node=c45f01n01-ib0.gpfs.net
rgName=BB1RGL daName=DA1 pdName=e4d5s03 pdLocation=SV13306129-5-3 pdFru=46W6911
pdWwn=naa.50 00C50055D4D437 pdState=dead/systemDrain
```

For more information about the `mmaddcallback` command, see the *IBM Storage Scale: Command and Programming Reference*.

IBM Storage Scale RAID events in syslog

If the Linux `syslog` facility is enabled, IBM Storage Scale RAID will log IBM Storage Scale RAID events to the syslog. This can be controlled using the IBM Storage Scale RAID `systemLogLevel` configuration variable. By default, all informational messages and higher-level error messages are logged.

Log events would look something like this:

```
Feb 27 15:43:20 c45f01n01 mmfsd: Error=MMFS_PDISK_FAILED, ID=0x157E5F74, Tag=1024872:
Pdisk Failed: Location=SV13306129-5-3, FRU=46W6911, WWID=5000c50055d4d437, RecoveryGrou
p=BB1RGL, DeclusteredArray=DA1, Pdisk=e4d5s03, PdiskState=dead/systemDrain
Feb 27 15:48:10 c45f01n01 mmfsd: Error=MMFS_PDISK_RECOVERED, ID=0x5DCF6F0A, Tag=1024873:
Pdisk Recovered: Location=SV13306129-5-3, FRU=46W6911, WWID=5000c50055d4d437, Recove
ryGroup=BB1RGL, DeclusteredArray=DA1, Pdisk=e4d5s03
```

Monitoring system health by using ESS GUI

The system provides background monitoring capabilities to check the health of a cluster and each node of the cluster, including all the services that are hosted on a node. You can view the system health states or corresponding events for the selected health state on the individual pages, widgets or panels of the ESS GUI. You can also view system health details by issuing the **`mmhealth`** command options like **`mmhealth cluster show`**, **`mmhealth node show`**, or other similar options.

The following table lists the system health monitoring options that are available in the ESS GUI.

Table 9. System health monitoring options that are available in ESS GUI

Option	Function
Home	Provides overall system health of the ESS system.
System overview widget in the **Monitoring** > **Dashboard** page	Displays the number of events that are reported against each component.
System health events widget in the **Monitoring** > **Dashboard** page	Provides an overview of the events that are reported in the system.
Timeline widget in the **Monitoring** > **Dashboard** page	Displays the events that are reported in a particular timeframe on the selected performance chart.
Filesets with the largest growth rate last week widget in the **Monitoring** > **Dashboard** page	Displays the filesets with the highest growth rate in the last one week.
File system capacity by fileset widget in the **Monitoring** > **Dashboard** page	Displays the capacity reported per fileset in a file system. The per fileset capacity data requires quota enablement at the file system level.
Monitoring > **Hardware**	Displays the status of all servers, enclosures, and drives in an enclosure. Click a server or enclosure to view the details of that particular hardware component.

Table 9. System health monitoring options that are available in ESS GUI (continued)	
Option	**Function**
Monitoring > **Hardware Details**	Displays status and details of the servers and enclosures that are a part of the system. Click any of the system components to view its details. This view also provides a text search and filtering for unhealthy hardware.
Monitoring > **Events**	Lists the events that are reported in the system. You can monitor and troubleshoot errors on your system from the **Events** page.
Monitoring > **Tips**	Lists the tips that are reported in the system and allows user to hide or show tips. The tip events give recommendations to the user to avoid certain issues that might occur in the future.
Monitoring > **Thresholds**	Lists the events that are raised when certain thresholds are reached for the data that is collected through performance monitoring sensors.
Monitoring > **Event Notifications**	Enables you to configure event notifications to notify the users about significant event changes that occur in the system.
Nodes	Lists the events that are reported at the node level.
Files > **File Systems**	Lists the events that are reported at the file system level.
Files > **Transparent Cloud Tiering**	Lists the events that are reported for the Transparent Cloud Tiering service. The GUI displays the page only when the transparent cloud tiering feature is enabled in the system.
Files > **Filesets**	Lists events that are reported for filesets.
Files > **Active File Management**	Displays health status and lists events that are reported for AFM cache relationship, AFM disaster recovery (AFMDR) relationship, and gateway nodes.
Storage > **Pools**	Displays health status and lists events that are reported for storage pools.
Storage > **NSDs**	Lists the events that are reported at the NSD level.
Storage > **Physical**	Displays health information and properties of physical disks and the corresponding declustered arrays.
Storage > **Volumes**	Displays health information and properties of logical volumes and the corresponding declustered arrays.

Note: The alerts and Tips icons on the ESS GUI header displays the number of tips and alerts that are received. It specifies the number and age of events that are triggered. The notifications disappear when the alert or tip is resolved.

Monitoring events by using GUI

You can primarily use the **Monitoring** > **Events** page to monitor the events that are reported in the system. The status bar that is placed on the upper right of the GUI header, also displays the number of events that are reported in the system.

The events are raised against the respective component, for example, GPFS, NFS, SMB, and others. Some of these events might occur multiple times in the system. Such events are grouped under the **Event Groups** tab and the number of occurrences of the events are indicated in the **Occurrences** column. The **Individual Events** tab lists all the events irrespective of the multiple occurrences.

A graphical view of events that are reported against each component is available. Clicking the graph displays only the relevant events in the grid view. Clicking a section on the graphical view applies the corresponding filter on the search action and fetches only the relevant data in the events table.

You can further filter the events that are listed in the **Events** page with the help of the following filter options:

- **Current Issues** displays all unfixed errors and warnings.
- **Current State** lists active events that are generated because of state changes.
- **Notices** displays the events that are not caused by a state change of an entity or component. Notices never become inactive on their own. Use the **Mark Notices as Read** action to make them historical when read.
- **All Events** displays all the events irrespective of severity and type. It shows both active and historical events. The historical events are displayed with a grey-colored icon.

The severity icons help to quickly determine whether the event is informational, a warning, or an error. Click an event and select **Properties** from the **Action** menu to see detailed information on the event. The event table displays the most recent events first.

The following table describes the severity levels of events.

Table 10. Notification levels

Notification level	Description
Error	Error notification is sent to indicate a problem that must be corrected as soon as possible. This notification indicates a serious problem with the system. For example, the event that is being reported might indicate a loss of redundancy in the system, and it is possible that another failure might result in loss of access to data.
Warning	A warning notification is sent to indicate a problem or unexpected condition with the system. Always immediately investigate this type of notification to determine the effect that it might have on your operation, and make any necessary corrections.
Information	An informational notification is sent to indicate the occurrence of an expected event. For example, a NAS service is started. No remedial action is required when these notifications are sent.

Note: A separate event type with severity "Tip" is also available. Tips are the recommendations that are given to ensure that you avoid certain issues that might occur in the future. The tip events are monitored separately in the **Monitoring** > **Tips** page of the GUI.

Marking events as read

The basic types of events can be categorized in the following manner.

- State change events - The type of event that is generated when an entity changes its state.

- The non-state change events - The type of event that is generated when there is no change in the entity's state.

The state change events are managed by the system and such events become inactive as soon as the state changes again. The non-state change events are referred to as notices in the GUI. Notices never become inactive on their own.

You must use the action **Mark Selected Notices as Read** or **Mark All Notices as Read** on the non-state change events. By using these actions, you can make them historical because the system displays those events as active events even if the problem or information is not valid anymore.

The result of *mark as read* operation is stored locally on the GUI node. That is, the changes that are made to the events are not visible through the other GUI nodes.

Resolving Event

Some issues can be resolved manually. To resolve events created for such issues, select the event and then click the **Resolve Event** option that is available under the **Actions** menu. On selecting the option, the **`mmhealth event resolve`** command is run to resolve the specific event. You can also right-click an event and select **Resolve Event** option from the drop-down menu that appears. On completion of the task, the status appears in the task window. The complete event thread can be viewed under the detailed view that you can access by using the **View Details** option.

Running fix procedure

Some issues can be resolved by running a fix procedure. To run a fix procedure, select **Run Fix Procedure** option that is available in the **Actions** menu.

Event notifications

The system can use Simple Network Management Protocol (SNMP) traps and emails to notify you when significant events are detected. Any combination of these notification methods can be used simultaneously. Use **Monitoring** > **Event Notifications** page in the GUI to configure event notifications.

Notifications are normally sent immediately after an event is raised.

In email notification method, you can also define whether a recipient needs to get a report of events that are reported in the system. These reports are sent only once in a day. Based on the seriousness of the issue, each event that is reported in the system gets a severity level that is associated with it.

The following table describes the severity levels of event notifications.

Table 11. Notification levels

Notification level	Description
Error	Error notification is sent to indicate a problem that must be corrected as soon as possible. This notification indicates a serious problem with the system. For example, the event that is being reported might indicate a loss of redundancy in the system, and it is possible that another failure might result in loss of access to data. The most typical reason that this type of notification is because of a hardware failure, but some configuration errors or fabric errors also are included in this notification level.
Warning	A warning notification is sent to indicate a problem or unexpected condition with the system. Always immediately investigate this type of notification to determine the effect that it might have on your operation, and make any necessary corrections. Therefore, a warning notification does not require any replacement parts and it does not require IBM Support Center involvement.

Table 11. Notification levels (continued)	
Notification level	**Description**
Information	An informational notification is sent to indicate that an expected event is occurred. For example, a NAS service is started. No remedial action is required when these notifications are sent.

Configuring email notifications

The email feature transmits operational and error-related data in the form of an event notification email.

To configure an email server, from the **Event Notifications** page, select **Email Server**. Select **Edit** and then click **Enable email notifications**. Enter required details and when you are ready, click **OK**.

Email notifications can be customized by setting a custom header and footer for the emails and customizing the subject by selecting and combining from the following variables: *&message, &messageId, &severity, &dateAndTime, &cluster*, and *&component*.

Emails containing the quota reports and other events that are reported in the following functional areas are sent to the recipients:

- AFM and AFM DR
- Authentication
- CES network
- Transparent Cloud Tiering
- NSD
- File system
- GPFS
- GUI
- Hadoop connector
- iSCSI
- Keystone
- Network
- NFS
- Object
- Performance monitoring
- SMB
- Object authentication
- Node
- CES

You can specify the severity level of events and whether to send a report that contains a summary of the events received.

To create email recipients, select **Email Recipients** from the **Event Notifications** page, and then click **Create Recipient**.

Note: You can change the email notification configuration or disable the email service at any time.

Configuring SNMP manager

Simple Network Management Protocol (SNMP) is a standard protocol for managing networks and exchanging messages. The system can send SNMP messages that notify personnel about an event. You can use an SNMP manager to view the SNMP messages that the system sends.

With an SNMP manager, such as IBM Systems Director, you can view and act on the messages that the SNMP agent sends. The SNMP manager can send SNMP notifications, which are also known as traps,

when an event occurs in the system. Select **Settings** > **Event Notifications** > **SNMP Manager** to configure SNMP managers for event notifications. You can specify up to a maximum of six SNMP managers.

Note: The GUI-based SNMP service supports only SNMP traps for system health events. SNMP queries are not supported.

In the SNMP mode of event notification, one SNMP notification (trap) with object identifiers (OID). 1.3.6.1.4.1.2.6.212.10.0.1 is sent by the GUI for each event. The following table provides the SNMP objects that are included in the event notifications.

Table 12. SNMP objects included in event notifications

OID	Description	Examples
1.3.6.1.4.1.2.6.212.10.1.1	Cluster ID	317908494245422510
1.3.6.1.4.1.2.6.212.10.1.2	Entity type	NODE, FILESYSTEM
1.3.6.1.4.1.2.6.212.10.1.3	Entity name	gss-11, fs01
1.3.6.1.4.1.2.6.212.10.1.4	Component	NFS, FILESYSTEM, NSD
1.3.6.1.4.1.2.6.212.10.1.5	Severity	INFO, TIP, WARNING, ERROR
1.3.6.1.4.1.2.6.212.10.1.6	Date and time	17.02.2016 13:27:42.516
1.3.6.1.4.1.2.6.212.10.1.7	Event name	nfs_active
1.3.6.1.4.1.2.6.212.10.1.8	Message	NFS service is now active.
1.3.6.1.4.1.2.6.212.10.1.9	Reporting node	The node where the problem is reported.

SNMP OID ranges

The following table gives the description of the SNMP OID ranges.

Table 13. SNMP OID ranges

OID range	Description
1.3.6.1.4.1.2.6.212	IBM Spectrum Scale
1.3.6.1.4.1.2.6.212.10	IBM Spectrum Scale GUI
1.3.6.1.4.1.2.6.212.10.0.1	IBM Spectrum Scale GUI event notification (trap)
1.3.6.1.4.1.2.6.212.10.1.x	IBM Spectrum Scale GUI event notification parameters (objects)

The traps for the core IBM Storage Scale and those trap objects are not included in the SNMP notifications that are configured through the IBM Storage Scale management GUI. .

Example for SNMP traps

The following example shows the SNMP event notification that is sent when performance monitoring sensor is shut down on a node:

```
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.2.6.212.10.0.1
SNMPv2-SMI::enterprises.2.6.212.10.1.1 = STRING: "317908494245422510"
SNMPv2-SMI::enterprises.2.6.212.10.1.2 = STRING: "NODE"
SNMPv2-SMI::enterprises.2.6.212.10.1.3 = STRING: "gss-11"
SNMPv2-SMI::enterprises.2.6.212.10.1.4 = STRING: "PERFMON"
SNMPv2-SMI::enterprises.2.6.212.10.1.5 = STRING: "ERROR"
SNMPv2-SMI::enterprises.2.6.212.10.1.6 = STRING: "18.02.2016 12:46:44.839"
SNMPv2-SMI::enterprises.2.6.212.10.1.7 = STRING: "pmsensors_down"
SNMPv2-SMI::enterprises.2.6.212.10.1.8 = STRING: "pmsensors service should be started and is
```

```
stopped"
SNMPv2-SMI::enterprises.2.6.212.10.1.9 = STRING: "gss-11"
```

The following example shows the SNMP event notification that is sent for an SNMP test message:

```
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.2.6.212.10.0.1
SNMPv2-SMI::enterprises.2.6.212.10.1.1 = STRING: "317908494245422510"
SNMPv2-SMI::enterprises.2.6.212.10.1.2 = STRING: "CLUSTER"
SNMPv2-SMI::enterprises.2.6.212.10.1.3 = STRING: "UNKNOWN"
SNMPv2-SMI::enterprises.2.6.212.10.1.4 = STRING: "GUI"
SNMPv2-SMI::enterprises.2.6.212.10.1.5 = STRING: "INFO"
SNMPv2-SMI::enterprises.2.6.212.10.1.6 = STRING: "18.02.2016 12:47:10.851"
SNMPv2-SMI::enterprises.2.6.212.10.1.7 = STRING: "snmp_test"
SNMPv2-SMI::enterprises.2.6.212.10.1.8 = STRING: "This is a SNMP test message."
SNMPv2-SMI::enterprises.2.6.212.10.1.9 = STRING: "gss-11"
```

SNMP MIBs

The SNMP Management Information Base (MIB) is a collection of definitions that define the properties of the managed objects.

The IBM Spectrum Scale GUI MIB OID range starts with 1.3.6.1.4.1.2.6.212.10. The OID range 1.3.6.1.4.1.2.6.212.10.0.1 denotes IBM Spectrum Scale GUI event notification (trap) and 1.3.6.1.4.1.2.6.212.10.1.x denotes IBM Spectrum Scale GUI event notification parameters (objects). While configuring SNMP, use the MIB file that is available at the following location of each GUI node: `/usr/lpp/mmfs/gui/IBM-SPECTRUM-SCALE-GUI-MIB.txt`.

Performance monitoring

The performance monitoring tool helps to view the metrics that are associated with GPFS and the associated protocols, get a graphical representation of the status and trends of the key performance indicators, and analyze ESS performance problems. You can use both CLI and GUI to monitor the system performance based on various aspects.

The following options are available to monitor system performance:

- **mmperfmon** command that helps to fetch system performance details that are collected through the performance monitoring tools.
- **mmpmon** command that helps to monitor the GPFS performance on the node in which it is run, and other specified nodes.
- ESS GUI
- Open source tool that is called Grafana can be used to monitor performance details that are collected through the performance monitoring tools.

For more information on how to monitor system performance by using mmperfmon and mmpmon commands and Grafana, see Performance monitoring documentation in *IBM Storage Scale: Problem Determination Guide*.

For more information on performance monitoring options that are available in GUI, see “Performance monitoring using ESS GUI” on page 94.

Performance monitoring using ESS GUI

The ESS GUI provides a graphical representation of the status and historical trends of the key performance indicators. The manner in which information is displayed on the GUI, helps Users to make quick and effective decisions, easily.

The following table lists the performance monitoring options that are available in the ESS GUI.

Table 14. Performance monitoring options available in ESS GUI	
Option	**Function**
Monitoring > **Statistics**	Displays performance of system resources and file and object storage in various performance charts. You can select the necessary charts and monitor the performance based on the filter criteria. The pre-defined performance widgets and metrics help in investigating every node or any particular node that is collecting the metrics.
Monitoring > **Dashboards**	Provides a more readable and real-time user interface that shows a graphical representation of the status and historical trends of key performance indicators. The dashboard view helps decision-making easier without wasting time.
Nodes	Provides an easy way to monitor the performance, health status, and configuration aspects of all available nodes in the ESS cluster.
Cluster > **Network**	Provides an easy way to monitor the performance and health status of various types of networks and network adapters.
Monitoring > **Thresholds**	Provides an option to configure and various thresholds based on the performance monitoring metrics. You can also monitor the threshold rules and the events that are associated with each rule.
Files > **File Systems**	Provides a detailed view of the performance, capacity, and health aspects of file systems.
Files > **Filesets**	Provides a detailed view of the fileset capacity.
Storage > **Pools**	Provides a detailed view of the performance, capacity, and health aspects of storage pools.
Storage > **NSDs**	Provides a detailed view of the performance, capacity, and health aspects of individual NSDs.
Protocols > **NFS Exports**	Provides an overview of the performance aspects of the NFS export.
Protocols > **SMB Shares**	Provides an overview of the performance aspects of the SMB shares.
Files > **Transparent Cloud Tiering**	Provides insight into health, performance, and configuration of the transparent cloud tiering service.
Files > **Active File Management**	Provides a detailed view of the configuration, performance, and health status of AFM cache relationship, AFM disaster recovery (AFMDR) relationship, and gateway nodes.

The **Statistics** page is used for selecting the attributes based on which the performance of the system needs to be monitored and comparing the performance based on the selected metrics. You can also mark charts as favorite charts and these charts become available for selection when you add widgets in the dashboard. You can display only two charts at a time in the **Statistics** page.

Favorite charts that are defined in the **Statistics** page and the predefined charts are available for selection in the **Dashboard**.

You can configure the system to monitor the performance of the following functional areas in the system:

- Network
- System resources
- Native RAID
- NSD server
- IBM Storage Scale client
- NFS
- SMB
- Object
- CTDB
- Transparent cloud tiering. This option is available only when the cluster is configured to work with the transparent cloud tiering service.
- Waiters
- AFM

Note: The functional areas such as NFS, SMB, Object, CTDB, and Transparent cloud tiering are available only if the feature is enabled in the system.

The performance and capacity data are collected with the help of the following two components:

Sensor
: The sensors are placed on all the nodes and they share the data with the collector. The sensors run on any node that is needed to collect metrics. Sensors are started by default on the protocol nodes.

Collector
: Collects data from the sensors. The metric collector runs on a single node and gathers metrics from all the nodes that are running the associated sensors. The metrics are stored in a database on the collector node. The collector ensures aggregation of data when the data gets older. The collector can run on any node in the system. By default, the collector runs on the management node. You can configure multiple collectors in the system. To configure performance monitoring through GUI, it is mandatory to configure a collector on each GUI node.

The following picture provides a graphical representation of the performance monitoring configuration for GUI.

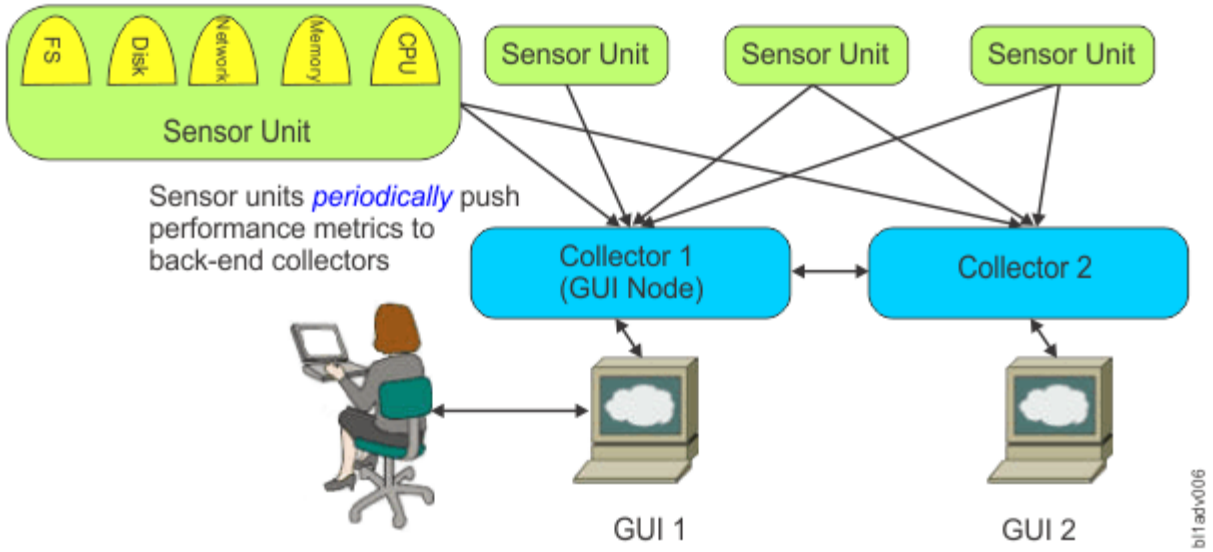


Figure 6. Performance monitoring configuration for GUI

You can use the **Services** > **Performance Monitoring** page to configure sensors. You can also use the **mmperfmon** command to configure the performance data collection through the CLI. The GUI displays a subset of the available metrics that are available in the performance monitoring tool.

Configuring performance monitoring options in GUI

You need to configure and enable the performance monitoring for GUI to view the performance data in the GUI.

Enabling performance tools in management GUI

You need to enable performance tools in the management GUI to display performance data in the management GUI. For more information, see *Enabling performance tools in management GUI* section in the *IBM Storage Scale: Administration Guide*.

Configuring capacity-related sensors to run on a single-node

Several capacity-related sensors must run only on a single node as they collect data for a clustered file system. For example, *GPFSDiskCap, GPFSFilesetQuota, GPFSFileset and GPFSPool*.

It is possible to automatically restrict these sensors to a single node. For new installations, capacity-related sensors are automatically configured to a single node where the capacity collection occurs. An updated cluster, which was installed before ESS 5.3.7 (IBM Storage Scale 5.0.5), might not be configured to use this feature automatically and must be reconfigured. To update the configuration, you can use the **mmperfmon config update SensorName.restrict=@CLUSTER_PERF_SENSOR** command, where **SensorName** values include *GPFSFilesetQuota, GPFSFileset, GPFSPool, and GPFSDiskCap*.

To collect file system and disk level capacity data on a single node that is selected by the system, run the following command to update the sensor configuration.

```
mmperfmon config update GPFSDiskCap.restrict=@CLUSTER_PERF_SENSOR
```

If the selected node is in the DEGRADED state, then the CLUSTER_PERF_SENSOR is automatically reconfigured to another node that is in the HEALTHY state. The performance monitoring service is restarted on the previous and currently selected nodes.

Note: If the *GPFSDiskCap* sensor is frequently restarted, it can negatively impact the system performance. The *GPFSDiskCap* sensor can cause a similar impact on the system performance as the **mmdf** command. Therefore, to avoid using the **@CLUSTER_PERF_SENSOR** for any sensor in the **restrict** field of a single node sensor until the node stabilizes in the HEALTHY state, it is advisable to use a dedicated healthy node. If you manually configure the **restrict** field of the capacity sensors then you must ensure that all the file systems on the specified node are mounted to record file system-related data, like capacity.

Use the **Services** > **Performance Monitoring** page to select the appropriate data collection periods for these sensors.

For the *GPFSDiskCap* sensor, the recommended period is 86400, which means once per day. As the *GPFSDiskCap.period* sensor runs **mmdf** command to get the capacity data, it is not recommended to use a value less than 10800 (every 3 hours). To show fileset capacity information, it is necessary to enable quota for all file systems where fileset capacity must be monitored. For more information, see the **-q** option in the **mmchfs** command and **mmcheckquota** command.

To update the sensor configuration for triggering an hourly collection of capacity-based fileset capacity information, run the **mmperfmon** command as shown in the following example,

```
mmperfmon config update GPFSFilesetQuota.restrict=@CLUSTER_PERF_SENSOR gui_node
GPFSFilesetQuota.period=3600
```

Verifying sensor and collector configurations

Do the following to verify whether collectors are working properly:

1. Issue **`systemctl status pmcollector`** on the GUI node to confirm that the collector is running. Start collector it if it is not started already.
2. If you cannot start the service, verify the log file that is located at the following location to fix the issue: `/var/log/zimon/ZIMonCollector.log`.
3. Use a sample CLI query to test if data collection works properly. For example,

```
mmperfmon query cpu_user
```

Do the following to verify whether sensors are working properly:

1. Confirm that the sensor is configured correctly by issuing the **`mmperfmon config show`** command. This command lists the content of the sensor configuration that is at the following location:`/opt/IBM/zimon/ZIMonSensors.cfg`. The configuration must point to the node where the collector is running and all the expected sensors must be enabled. An enabled sensor has a period greater than `0` in the same config file.
2. Issue **`systemctl status pmsensors`** to verify the status of the sensors.

Configuring performance metrics and display options in the Statistics page of the GUI

Use the **Monitoring** > **Statistics** page to monitor the performance of system resources and file and object storage. Performance of the system can be monitored by using various pre-defined charts. You can select the required charts and monitor the performance based on the filter criteria.

The pre-defined performance charts and metrics help in investigating every node or any particular node that is collecting the metrics. The following figure shows various configuration options that are available in the **Statistics** page of the management GUI.

Figure 7. Statistics page in the IBM Storage Scale management GUI

You can select pre-defined charts that are available for selection from pre-defined chart list. You can display up to two charts at a time.

Display options in performance charts

The charting section displays the performance details based on various aspects. The GUI provides a rich set of controls to view performance charts. You can use these controls to perform the following actions on the charts that are displayed on the page:

- Zoom the chart by using the mouse wheel or resizing the timeline control. Y-axis is automatically adjusted during zooming.
- Click and drag the chart or the timeline control at the bottom. Y-axis is automatically adjusted during panning.
- Compare charts side by side. You can synchronize the y-axis and bind the x-axis. To modify the X and Y axes of the chart, click the configuration symbol next to the title *Statistics* and select the required options.
- Link the timelines of the two charts together by using the display options that are available.
- The Dashboard helps to access all single graph charts, which are either predefined or custom created favorites.

Selecting performance and capacity metrics

To monitor the performance of the system, you need to select the appropriate metrics to be displayed in the performance charts. Metrics are grouped under the combination of resource types and aggregation levels. The resource types determine the area from which the data is taken to create the performance analysis and aggregation level determines the level at which the data is aggregated. The aggregation levels that are available for selection varies based on the resource type.

Sensors are configured against each resource type. The following table provides a mapping between resource types and sensors under the Performance category.

Table 15. Sensors available for each resource type

Resource type	**Sensor name**	**Candidate nodes**
Network	Network	All nodes
System Resources	CPU	All nodes
	Load	
	Memory	
NSD Server	GPFSNSDDisk	NSD server nodes
IBM Storage Scale Client	GPFSFilesystem	IBM Storage Scale Client nodes
	GPFSVFS	
	GPFSFilesystemAPI	
Native RAID	GPFSPDDisk	Elastic Storage Server (ESS) building block nodes
NFS	NFSIO	Protocol nodes running NFS service
SMB	SMBStats	Protocol nodes running SMB service
	SMBGlobalStats	
CTDB	CTDBStats	Protocol nodes running SMB service

Table 15. Sensors available for each resource type (continued)		
Resource type	**Sensor name**	**Candidate nodes**
Waiters	GPFSWaiters	All nodes
Object	SwiftAccount	Protocol nodes running Object service
	SwiftContainer	
	SwiftObject	
	SwiftProxy	
AFM	GPFSAFM	AFM gateway nodes
	GPFSAFMFS	
	GPFSAFMFSET	
Transparent Cloud Tiering	MCStoreGPFSStats	Cloud gateway nodes
	MCStoreIcstoreStats	
	MCStoreLWEStats	

The resource type *Waiters* are used to monitor the long running file system threads. Waiters are characterized by the purpose of the corresponding file system threads. For example, an RPC call waiter that is waiting for Network I/O threads or a waiter that is waiting for a local disk I/O file system operation. Each waiter has a wait time associated with it and it defines how long the waiter is already waiting. With some exceptions, long waiters typically indicate that something in the system is not healthy.

The *Waiters* performance chart shows the aggregation of the total count of waiters of all nodes in the cluster above a certain threshold. Different thresholds from 100 milliseconds to 60 seconds can be selected in the list below the aggregation level. By default, the value shown in the graph is the sum of the number of waiters that exceed threshold in all nodes of the cluster at that point in time. The filter functionality can be used to display waiters data only for some selected nodes or file systems. Furthermore, there are separate metrics for different waiter types such as Local Disk I/O, Network I/O, ThCond, ThMutex, Delay, and Syscall.

You can also monitor the capacity details that are aggregated at the following levels:

- NSD
- Node
- File system
- Pool
- Fileset
- Cluster

The following table lists the sensors that are used for capturing the capacity details.

Table 16. Sensors available to capture capacity details	
Sensor name	**Candidate nodes**
DiskFree	All nodes
GPFSFilesetQuota	Only a single node
GPFSDiskCap	Only a single node
GPFSPool	Only a single node where all GPFS file systems are mounted.
GPFSFileset	Only a single node.

You can edit an existing chart by clicking the ellipsis icon on the performance chart header and select **Edit** to modify the metrics selections. Follow the steps shown to drill down to the relevant metric:

1. Select the cluster to be monitored from the **Cluster** field. You can either select the local cluster or the remote cluster.
2. Select **Resource type**. This is the area from which the data is taken to create the performance analysis.
3. Select **Aggregation level**. The aggregation level determines the level at which the data is aggregated. The aggregation levels that are available for selection varies based on the resource type.
4. Select the entities that need to be graphed. The table lists all entities that are available for the chosen resource type and aggregation level. When a metric is selected, you can also see the selected metrics in the same grid and use methods like sorting, filtering, or adjusting the time frame to select the entities that you want to select.
5. Select **Metrics**. Metrics is the type of data that need to be included in the performance chart. The list of metrics that is available for selection varies based on the resource type and aggregation type.
6. Use the **Filter** option to further narrow down the selection. Depending on the selected object category and aggregation level, the **Filter** section can be displayed underneath the aggregation level, allowing one or more filters to be set. Filters are specified as regular expressions as shown in the following examples:
   - As a single entity:

     node1

     eth0
   - Filter metrics applicable to multiple nodes as shown in the following examples:
     - To select a range of nodes such as node1, node2 and node3:

       node1|node2|node3

       node[1-3]
     - To filter based on a string of text. For example, all nodes starting with 'nod' or ending with 'int':

       nod.+|.+int
     - To filter network interfaces eth0 through eth6, bond0 and eno0 through eno6:

       eth[0-6]|bond0|eno[0-6]
     - To filter nodes starting with 'strg' or 'int' and ending with 'nx':

       (strg)|(int).+nx

Creating favorite charts

Favorite charts are nothing but customized predefined charts. Favorite charts along with the predefined charts are available for selection when you add widgets in the **Dashboard** page.

To create favorite charts, click the ‘star’ symbol that is placed next to the chart title and enter the label.

Configuring the dashboard to view performance charts

The **Monitoring** > **Dashboard** page provides an easy to read, single page, and real-time user interface that provides a quick overview of the system performance.

The dashboard consists of several dashboard widgets and the associated favorite charts that can be displayed within a chosen layout. Currently, the following important widget types are available in the dashboard:

- Performance
- File system capacity by fileset
- System health events

- System overview
- Filesets with the largest growth rate in last week
- Timeline

The following picture highlights the configuration options that are available in the edit mode of the dashboard.

Figure 8. Dashboard page in the edit mode

Layout options

The highly customizable dashboard layout options helps to add or remove widgets and change its display options. Select **Layout Options** option from the menu that is available in the upper right corner of the **Dashboard** GUI page to change the layout options. While selecting the layout options, you can either select the basic layouts that are available for selection or create a new layout by selecting an empty layout as the starting point.

You can also save the dashboard so that it can be used by other users. Select **Create Dashboard** and **Delete Dashboard** options from the menu that is available in the upper right corner of the **Dashboard** page to create and delete dashboards respectively. If several GUIs are running by using CCR, saved dashboards are available on all nodes.

When you open the IBM Storage Scale GUI after the installation or upgrade, you can see the default dashboards that are shipped with the product. You can further modify or delete the default dashboards to suit your requirements.

Widget options

Several dashboard widgets can be added in the selected dashboard layout. Select **Edit Widgets** option from the menu that is available in the upper right corner of the **Dashboard** GUI page to edit or remove widgets in the dashboard. You can also modify the size of the widget in the edit mode. Use the **Add Widget** option that is available in the edit mode to add widgets in the dashboard.

The widgets with type *Performance* lists the charts that are marked as favorite charts in the **Statistics** page of the GUI. Favorite charts along with the predefined charts are available for selection when you add widgets in the dashboard.

To create favorite charts, click the ‘star’ symbol that is placed next to the chart title in the **Monitoring** > **Statistics** page.

Querying performance data shown in the GUI through CLI

You can query the performance data that is displayed in the GUI through the CLI. This is usually used for external system integration or to troubleshoot any issues with the performance data displayed in the GUI.

The following example shows how to query the performance data through CLI:

```
# mmperfmon query "sum(netdev_bytes_r)"
```

This query displays the following output:

```
Legend:
 1:     mr-31.localnet.com|Network|eth0|netdev_bytes_r
 2:     mr-31.localnet.com|Network|eth1|netdev_bytes_r
 3:     mr-31.localnet.com|Network|lo|netdev_bytes_r

Row           Timestamp netdev_bytes_r netdev_bytes_r netdev_bytes_r
  1 2016-03-15-14:52:09          10024
  2 2016-03-15-14:52:10           9456
  3 2016-03-15-14:52:11           9456
  4 2016-03-15-14:52:12           9456
  5 2016-03-15-14:52:13           9456
  6 2016-03-15-14:52:14           9456
  7 2016-03-15-14:52:15          27320
  8 2016-03-15-14:52:16           9456
  9 2016-03-15-14:52:17           9456
 10 2016-03-15-14:52:18          11387
```

The sensor gets the performance data for the collector and the collector passes it to the performance monitoring tool to display it in the CLI and GUI. If sensors and collectors are not enabled in the system, the system does not display the performance data and when you try to query data from a system resource, it returns an error message. For example, if performance monitoring tools are not configured properly for the resource type *Transparent Cloud Tiering*, the system displays the following output while querying the performance data:

```
mmperfmon query "sum(mcs_total_requests)" number_buckets 1
Error: No data available for query: 3169

mmperfmon: Command failed. Examine previous error messages to determine cause.
```

Monitoring performance of nodes

The **Monitoring** > **Nodes** page provides an easy way to monitor the performance, health status, and configuration aspects of all available nodes in the IBM Storage Scale cluster.

The **Nodes** page provides the following options to analyze performance of nodes:

1. A quick view that gives the number of nodes in the system, and the overall performance of nodes based on CPU and memory usages.

   You can access this view by selecting the expand button that is placed next to the title of the page. You can close this view if not required.

   The graphs in the overview show the nodes that have the highest average performance metric over a past period. These graphs are refreshed regularly. The refresh intervals of the top three entities are depended on the displayed time frame as shown:

   - Every minute for the 5 minutes time frame
   - Every 15 minutes for the 1 hour time frame
   - Every six hours for the 24 hours time frame
   - Every two days for the 7 days' time frame
   - Every seven days for the 30 days' time frame

- Every four months for the 365 days' time frame

2. A nodes table that displays many different performance metrics.

   To find nodes with extreme values, you can sort the values displayed in the nodes table by different performance metrics. Click the performance metric in the table header to sort the data based on that metric.

   You can select the time range that determines the averaging of the values that are displayed in the table and the time range of the charts in the overview from the time range selector, which is placed in the upper right corner. The metrics in the table do not update automatically. The refresh button at the top of the table allows to refresh the table content with more recent data.

   You can group the nodes to be monitored based on the following criteria:

   - All nodes
   - NSD server nodes
   - Protocol nodes

3. A detailed view of the performance and health aspects of individual nodes that are listed in the **Nodes** page.

   Select the node for which you need to view the performance details and select **View Details**. The system displays various performance charts on the right pane.

   The detailed performance view helps to drill-down to various performance aspects. The following list provides the performance details that can be obtained from each tab of the performance view:

   - **Overview** tab provides performance chart for the following:
     - Client IOPS
     - Client data rate
     - Server data rate
     - Server IOPS
     - Network
     - CPU
     - Load
     - Memory
   - **Events** tab helps to monitor the events that are reported in the node. Three filter options are available to filter the events by their status; such as **Current Issues**, **Unread Messages**, and **All Events** displays every event, no matter if it is fixed or marked as read. Similar to the **Events** page, you can also perform the operations like marking events as read and running fix procedure from this events view.
   - **File Systems** tab provides performance details of the file systems mounted on the node. You can view the file system read or write throughput, average read or write transactions size, and file system read or write latency.
   - **NSDs** tab gives status of the disks that are attached to the node. The NSD tab appears only if the node is configured as an NSD server.
   - **SMB** and **NFS** tabs provide the performance details of the SMB and NFS services hosted on the node. These tabs appear in the chart only if the node is configured as a protocol node.
   - **Network** tab displays the network performance details.

Monitoring performance of file systems

The **File Systems** page provides an easy way to monitor the performance, health status, and configuration aspects of the all available file systems in the ESS cluster.

The following options are available to analyze the file system performance:

1. A quick view that gives the number of protocol nodes, NSD servers, and NSDs that are part of the available file systems that are mounted on the GUI server. It also provides overall capacity and total throughput details of these file systems. You can access this view by selecting the expand button that is placed next to the title of the page. You can close this view if not required.

   The graphs displayed in the quick view are refreshed regularly. The refresh intervals are depended on the displayed time frame as shown:

   - Every minute for the 5 minutes time frame
   - Every 15 minutes for the 1 hour time frame
   - Every six hours for the 24 hours time frame
   - Every two days for the 7 days time frame
   - Every seven days for the 30 days time frame
   - Every four months for the 365 days time frame
2. A file systems table that displays many different performance metrics. To find file systems with extreme values, you can sort the values displayed in the file systems table by different performance metrics. Click the performance metric in the table header to sort the data based on that metric. You can select the time range that determines the averaging of the values that are displayed in the table and the time range of the charts in the overview from the time range selector, which is placed in the upper right corner. The metrics in the table do not update automatically. The refresh button at the top of the table allows to refresh the table with more recent data.
3. A detailed view of the performance and health aspects of individual file systems. To see the detailed view, you can either double-click on the file system for which you need to view the details or select the file system and click **View Details**.

   The detailed performance view helps to drill-down to various performance aspects. The following list provides the performance details that can be obtained from each tab of the performance view:

   - **Overview**: Provides an overview of the file system, performance, and properties.
   - **Events**: System health events reported for the file system.
   - **NSDs**: Details of the NSDs that are part of the file system.
   - **Pools**: Details of the pools that are part of the file system.
   - **Nodes**: Details of the nodes on which the file system is mounted.
   - **Filesets**: Details of the filesets that are part of the file system.
   - **NFS**: Details of the NFS exports created in the file system.
   - **SMB**: Details of the SMB shares created in the file system.
   - **Object**: Details of the ESS object storage on the file system.

Monitoring performance of NSDs

The NSDs page provides an easy way to monitor the performance, health status, and configuration aspects of the all network shared disks (NSD) that are available in the ESS cluster.

The following options are available in the NSDs page to analyze the NSD performance:

1. An NSD table that displays the available NSDs and many different performance metrics. To find NSDs with extreme values, you can sort the values that are displayed in the table by different performance metrics. Click the performance metric in the table header to sort the data based on that metric. You can select the time range that determines the averaging of the values that are displayed in the table from the time range selector, which is placed in the upper right corner. The metrics in the table are refreshed based on the selected time frame. You can refresh it manually to see the latest data.
2. A detailed view of the performance and health aspects of individual NSDs are also available in the NSDs page. Select the NSD for which you need to view the performance details and select **View Details.** The system displays various performance charts on the right pane.

The detailed performance view helps to drill-down to various performance aspects. The following list provides the performance details that can be obtained from each tab of the performance view:

- **Overview:** Provides an overview of the NSD performance details and related attributes.
- **Events:** System health events reported for the NSD.
- **Nodes:** Details of the nodes that serve the NSD.
- **Vdisk:** Displays the ESS aspects of NSD.

Monitoring IBM Storage Scale RAID I/O performance

You can use the IBM Storage Scale `mmpmon` command to monitor IBM Storage Scale RAID I/O performance.

The `mmpmon` command includes input requests for displaying and resetting vdisk I/O statistics.

For more information about the `mmpmon` command, see the *IBM Storage Scale: Command and Programming Reference*.

For more information about using `mmpmon` to monitor IBM Storage Scale I/O performance, see the *IBM Storage Scale: : Administration Guide*.

Displaying vdisk I/O statistics

To display vdisk I/O statistics, run `mmpmon` with the following command included in the input file:

```
vio_s [f [rg RecoveryGroupName [ da DeclusteredArrayName [ v VdiskName]]]] [reset]
```

This request returns strings containing vdisk I/O statistics as seen by that node. The values are presented as total values for the node, or they can be filtered with the `f` option. The reset option indicates that the statistics should be reset after the data is sampled.

If the `-p` option is specified when running `mmpmon`, the vdisk I/O statistics are provided in the form of keywords and values in the `vio_s` response. Table 17 on page 106 lists and describes these keywords in the order in which they appear in the output.

Table 17. Keywords and descriptions of values provided in the `mmpmon vio_s` response

Keyword	Description
`_n_`	The IP address of the node that is responding. This is the address by which IBM Storage Scale knows the node.
`_nn_`	The name by which IBM Storage Scale knows the node.
`_rc_`	The reason or error code. In this case, the reply value is 0 (OK).
`_t_`	The current time of day in seconds (absolute seconds since Epoch (1970)).
`_tu_`	The microseconds part of the current time of day.
`_rg_`	The name of the recovery group.
`_da_`	The name of the declustered array.
`_v_`	The name of the disk.
`_r_`	The total number of read operations.
`_sw_`	The total number of short write operations.
`_mw_`	The total number of medium write operations.
`_pfw_`	The total number of promoted full track write operations.
`_ftw_`	The total number of full track write operations.

Table 17. Keywords and descriptions of values provided in the mmpmon vio_s response (continued)

Keyword	Description
fuw	The total number of flushed update write operations.
fpw	The total number of flushed promoted full track write operations.
_m_	The total number of migrate operations.
_s_	The total number of scrub operations.
_l_	The total number log write operations.
_fc_	The total number of force consistency operations.
fix	The total number of buffer range fix operations.
ltr	The total number of log tip read operations.
lhr	The total number of log home read operations.
rgd	The total number of recovery group descriptor write operations.
meta	The total number metadata block write operations.

To display these statistics, use the sample script `/usr/lpp/mmfs/samples/vdisk/viostat`. The following shows the usage of the `viostat` script:

```
viostat  [-F NodeFile | [--recovery-group RecoveryGroupName
                         [--declustered-array  DeclusteredArrayName
                         [--vdisk VdiskName]]]]
         [Interval [Count]]
```

Example of mmpmon vio_s request

Suppose `commandFile` contains this line:

```
vio_s
```

and this command is issued:

```
mmpmon -i commandFile
```

The output is similar to this:

```
mmpmon node 172.28.213.53 name kibgpfs003 vio_s OK VIOPS per second
timestamp:                          1399010914/597072
recovery group:                     *
declustered array:                  *
vdisk:                              *
client reads:                       207267
client short writes:                26162
client medium writes:               2
client promoted full track writes:  1499
client full track writes:           864339
flushed update writes:              0
flushed promoted full track writes: 0
migrate operations:                 24
scrub operations:                   5307
log writes:                         878084
force consistency operations:       0
fixit operations:                   0
logTip read operations:             48
logHome read operations:            52
rgdesc writes:                      12
metadata writes:                    5153
```

Resetting vdisk I/O statistics

The **vio_s_reset** request resets the statistics that are displayed with **vio_s** requests.

Table 18 on page 108 describes the keywords for the `vio_s_reset` response, in the order that they appear in the output. These keywords are used only when `mmpmon` is invoked with the `-p` flag. The response is a single string.

*Table 18. Keywords and values for the **mmpmon vio_s_reset** response*

Keyword	Description
_n_	IP address of the node that is responding. This is the address by which IBM Storage Scale knows the node.
_nn_	The hostname that corresponds to the IP address (the _n_ value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Example of mmpmon vio_s_reset request

Suppose `commandFile` contains this line:

```
vio_s_reset
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_vio_s_reset_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431
```

If the `-p` flag is not specified, the output is similar to this:

```
mmpmon node 199.18.1.8 name node1 reset OK
```

Monitoring capacity through GUI

You can monitor the capacity of the file system, pools, filesets, NSDs, users, and user groups in the IBM Storage Scale system.

The capacity details displayed in the GUI are obtained from the following sources:

- GPFS quota database. The system collects the quota details and stores it in the PostgreSQL database.
- Performance monitoring tool collects the capacity data. The GUI queries the performance monitoring tool and displays capacity data in various pages in the GUI.

Based on the source of the capacity information, different procedures need to be performed to enable capacity and quota data collection.

For both GPFS quota database and performance monitoring tool-based capacity and quota collection, you need to use the **Files** > **Quotas** page to enable quota data collection per file system and enforce quota limit checking. If quota is not enabled for a file system:

- No capacity and inode data is collected for users, groups, and filesets.
- Quota limits for users, groups, and filesets cannot be defined.
- No alerts are sent and the data writes are not restricted.

To enable capacity data collection from the performance monitoring tool, the **GPFSFilesetQuota** sensor must be enabled. For more information on how to enable the performance monitoring sensor for capacity data collection, see *Manual installation of IBM Storage Scale GUI* in *IBM Storage Scale: Concepts, Planning, and Installation Guide*.

Capacity data obtained from the GPFS quota database

The capacity and quota information collected from the GPFS quota database is displayed on the Files > Quotas and Files > User Capacity pages in the management GUI.

1. **Files** > **Quotas** page

Use quotas to control the allocation of files and data blocks in a file system. You can create default, user, group, and fileset quotas through the **Quotas** page.

A quota is the amount of disk space and the amount of metadata that is assigned as upper limits for a specified user, group of users, or fileset. Use the **Actions** menu to create or modify quotas. The management GUI allows you to only manage capacity-related quota. The inode-related quota management is only possible in the command-line interface.

You can specify a soft limit, a hard limit, or both. When you set a soft limit quota, a warning is sent to the administrator when the file system is close to reaching its storage limit. A grace period starts when the soft quota limit is reached. Data is written until the grace period expires, or until the hard quota limit is reached. Grace time resets when used capacity goes less than the soft limit. If you set a hard limit quota, then you cannot save data after the quota is reached. If the quota is exceeded, then you must delete files or raise the quota limit to store more data.

Note:

- User or user group quotas for filesets are only supported if the *Per Fileset* option is enabled at the file system level. Use the command-line interface to set the option. See the man pages of **mmcrfs** and **mmchfs** commands for more detail.
- You need to unmount a file system to change the quota enablement method from per file system to per fileset or vice versa.

You can set default user quotas at the file system level rather than defining user quotas explicitly for each user. Default quota limits can be set for users. You can specify the general quota collection scope such as per file system or per fileset to define whether the default quota needs to be defined at file system level or fileset level and set the default user quota. After this value is set, all child objects that are created under the file system or fileset are configured with the default soft and hard limits. You can assign a custom quota limit to individual child objects, but the default limits remain the same unless changed at the file system or fileset level.

After reconfiguring quota settings, it is recommended to run the **mmcheckquota** command for the affected file system to verify the changes.

For more information on how to manage quotas, see *Managing GPFS quotas* section in the *IBM Storage Scale: Administration Guide*.

Capacity data from users, groups, and filesets with no quota limit set are not listed in the Quotas page. Use the **Files** > **User Capacity** page to see capacity information of such users and groups. Use the **Files** > **Filesets** page to view current and historic capacity information of filesets.

2. **Files** > **User Capacity** page

The **Files** > **User Capacity** page provides predefined capacity reports for users and groups. While capacity information of file systems, pools, and filesets is available in the respective areas of the GUI, the **Files** > **User Capacity** page is the only place where information on used capacity per user or group is available.

The User Capacity page depends on the quota accounting method of the file system. You need to enable quota for a file system to display the user capacity data. If quota is not enabled, you can follow the fix procedure in the **Files** > **Quotas** page or use the **mmchfs <Device> -Q yes** CLI command to enable quota. Even if the capacity limits are not set, the User Capacity page shows data as soon as the quota accounting is enabled and users write data. This is different in the Quotas page, where only

users and groups with quota limits defined are listed. The user and group capacity quota information is automatically collected once a day by the GUI.

For users and user groups, you can see the total capacity and whether quotas are set for these objects. you can also see the percentage of soft limit and hard limit usage. When the hard limit is exceeded, no more files belong to the respective user, user group, or fileset can be written. However, exceeding the hard limit allows a certain grace period before disallowing more file writes. Soft and hard limits for disk capacity are measured in units of kilobytes (KiB), megabytes (MiB), or gigabytes (GiB). Use the **Files** > **Quotas** page to change the quota limits.

Capacity data collected through the performance monitoring tool

The historical capacity data collection for file systems, pools, and filesets depend on the correctly configured data collection sensors for fileset quota and disk capacity. When the IBM Storage Scale system is installed through the installation toolkit, the capacity data collection is configured by default. In other cases, you need to enable capacity sensors manually.

If the capacity data collection is not configured correctly you can use **mmperfmon** CLI command or the **Services** > **Performance Monitoring** > **Sensors** page.

The **Services** > **Performance Monitoring** > **Sensors** page allows to view and edit the sensor settings. By default, the collection periods of capacity collection sensors are set to collect data with a period of up to one day. Therefore, it might take a while until the data is refreshed in the GUI.

The following sensors are collecting capacity related information and are used by the GUI.

GPFSDiskCap
: NSD, Pool and File system level capacity. Uses the **mmdf** command in the background and typically runs once per day as it is resource intensive. Should be restricted to run on a single node only.

GPFSPool
: Pool and file system level capacity. Requires a mounted file system and typically runs every 5 minutes. Should be restricted to run on a single node only.

GPFSFilesetQuota
: Fileset capacity based on the quota collection mechanism. Typically, runs every hour. Should be restricted to run only on a single node.

GPFSFileset
: Inode space (independent fileset) capacity and limits. Typically runs every 5 minutes. Should be restricted to run only on a single node.

DiskFree
: Overall capacity and local node capacity. Can run on every node.

The **Monitoring** > **Statistics** page allows to create customized capacity reports for file systems, pools and filesets. You can store these reports as favorites and add them to the dashboard as well.

The dedicated GUI pages combine information about configuration, health, performance, and capacity in one place. The following GUI pages provide the corresponding capacity views:

- **Files** > **File Systems**
- **Files** > **Filesets**
- **Storage** > **Pools**
- **Storage** > **NSDs**

The Filesets grid and details depend on quota that is obtained from the GPFS quota database and the performance monitoring sensor *GPFSFilesetQuota*. If quota is disabled, the system displays a warning dialog in the Filesets page.

Troubleshooting issues with capacity data displayed in the GUI

Due to the impact that capacity data collection can have on the system, different capacity values are collected on a different schedule and are provided by different system components. The following list

provides insight on the issues that can arise from the multitude of schedules and subsystems that provide capacity data:

Capacity in the file system view and the total amount of the capacity for pools and volumes view do not match.

The capacity data in the file system view is collected every 10 minutes by performance monitoring collector, but the capacity data for pools and Network Shared Disks (NSD) are not updated. By default, NSD data is only collected once per day by performance monitoring collector and it is cached. Clicking the refresh icon gathers the last two records from performance monitoring tool and it displays the last record values if they are not null. If the last record has null values, the system displays the previous one. If the values of both records are null, the system displays N/A and the check box for displaying a time chart is disabled. The last update date is the record date that is fetched from performance monitoring tool if the values are not null.

Capacity in the file system view and the total amount of used capacity for all filesets in that file system do not match.

There are differences both in the collection schedule as well as in the collection mechanism that contributes to the fact that the fileset capacities do not add up to the file system used capacity.

Scheduling differences:

Capacity information that is shown for filesets in the GUI is collected once per hour by performance monitoring collector and displayed on Filesets page. When you click the refresh icon you get the information of the last record from performance monitoring. If the last two records have null values, you get a 'Not collected' warning for used capacity. The file system capacity information on the file systems view is collected every 10 minutes by performance monitoring collector and when you click the refresh icon you get the information of the last record from performance monitoring.

Data collection differences:

Quota values show the sum of the size of all files and are reported asynchronously. The quota reporting does not consider metadata, snapshots, or capacity that cannot be allocated within a subblock. Therefore, the sum of the fileset quota values can be lower than the data shown in the file system view. You can use the CLI command **mmlsfileset** with the **-d** and **-i** options to view capacity information. The GUI does not provide a means to display these values because of the performance impact due to data collection.

The sum of all fileset inode values on the view quota window does not match the number of inodes that are displayed on the file system properties window.

The quota value only accounts for user-created inodes while the properties for the file system also display inodes that are used internally. Refresh the quota data to update these values.

No capacity data shown on a new system or for a newly created file system

Capacity data may show up with a delay of up to 1 day. The capacity data for file systems, NSDs, and pools is collected once a day as this is a resource intensive operation. Line charts do not show a line if only a single data point exists. You can use the hover function in order to see the first data point in the chart.

The management GUI displays negative fileset capacity or an extremely high used capacity like millions of Petabytes or 4000000000 used inodes.

This problem can be seen in the quota and filesets views. This problem is caused when the quota accounting is out of sync. To fix this error, issue the **mmcheckquota** command. This command recounts inode and capacity usage in a file system by user, user group, and fileset, and writes the collected data into the database. It also checks quota limits for users, user groups, and filesets in a file system. Running this command can impact performance of I/O operations.

No capacity data is displayed on the performance monitoring charts

Verify whether the GPFSFilesetQuota sensor is enabled. You can check the sensor status form the Services > Performance Monitoring page in the GUI. For more information on how to enable the performance monitoring sensor for capacity data collection, see *Manual installation of IBM Storage Scale GUI* in *IBM Storage Scale: Concepts, Planning, and Installation Guide*.

Chapter 9. Checking the health of an ESS configuration: a sample scenario

The scenario presented here shows how to use the `gnrhealthcheck` sample script to check the general health of an ESS configuration.

1. In this example, all checks are successful.

   To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   The system displays information similar to this:

   ```
   ################################################################################
   # Beginning topology checks.
   ################################################################################
   Topology checks successful.

   ################################################################################
   # Beginning enclosure checks.
   ################################################################################
   Enclosure checks successful.

   ################################################################################
   # Beginning recovery group checks.
   ################################################################################
   Recovery group checks successful.

   ################################################################################
   # Beginning pdisk checks.
   ################################################################################
   Pdisk group checks successful.

   ########################################
   Beginning IBM Power RAID checks.
   ########################################
   IBM Power RAID checks successful.
   ```

2. In this example, several issues need to be investigated.

   To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   The system displays information similar to this:

   ```
   ################################################################################
   # Beginning topology checks.
   ################################################################################
   Found topology problems on node c45f01n01-ib0.gpfs.net

   DCS3700 enclosures found: 0123456789AB SV11812206 SV12616296 SV13306129
   Enclosure 0123456789AB (number 1):
   Enclosure 0123456789AB ESM A sg244[0379][scsi8 port 4] ESM B sg4[0379][scsi7 port 4]
   Enclosure 0123456789AB Drawer 1 ESM sg244 12 disks diskset "19968" ESM sg4 12 disks diskset "19968"
   Enclosure 0123456789AB Drawer 2 ESM sg244 12 disks diskset "11294" ESM sg4 12 disks diskset "11294"
   Enclosure 0123456789AB Drawer 3 ESM sg244 12 disks diskset "60155" ESM sg4 12 disks diskset "60155"
   Enclosure 0123456789AB Drawer 4 ESM sg244 12 disks diskset "03345" ESM sg4 12 disks diskset "03345"
   Enclosure 0123456789AB Drawer 5 ESM sg244 11 disks diskset "33625" ESM sg4 11 disks diskset "33625"
   Enclosure 0123456789AB sees 59 disks

   Enclosure SV12616296 (number 2):
   Enclosure SV12616296 ESM A sg63[0379][scsi7 port 3] ESM B sg3[0379][scsi9 port 4]
   Enclosure SV12616296 Drawer 1 ESM sg63 11 disks diskset "51519" ESM sg3 11 disks diskset "51519"
   Enclosure SV12616296 Drawer 2 ESM sg63 12 disks diskset "36246" ESM sg3 12 disks diskset "36246"
   Enclosure SV12616296 Drawer 3 ESM sg63 12 disks diskset "53750" ESM sg3 12 disks diskset "53750"
   ```

```
Enclosure SV12616296 Drawer 4 ESM sg63 12 disks diskset "07471" ESM sg3 12 disks diskset "07471"
Enclosure SV12616296 Drawer 5 ESM sg63 11 disks diskset "16033" ESM sg3 11 disks diskset "16033"
Enclosure SV12616296 sees 58 disks

Enclosure SV11812206 (number 3):
Enclosure SV11812206 ESM A sg66[0379][scsi9 port 3] ESM B sg6[0379][scsi8 port 3]
Enclosure SV11812206 Drawer 1 ESM sg66 11 disks diskset "23334" ESM sg6 11 disks diskset "23334"
Enclosure SV11812206 Drawer 2 ESM sg66 12 disks diskset "16332" ESM sg6 12 disks diskset "16332"
Enclosure SV11812206 Drawer 3 ESM sg66 12 disks diskset "52806" ESM sg6 12 disks diskset "52806"
Enclosure SV11812206 Drawer 4 ESM sg66 12 disks diskset "28492" ESM sg6 12 disks diskset "28492"
Enclosure SV11812206 Drawer 5 ESM sg66 11 disks diskset "24964" ESM sg6 11 disks diskset "24964"
Enclosure SV11812206 sees 58 disks

Enclosure SV13306129 (number 4):
Enclosure SV13306129 ESM A sg64[0379][scsi8 port 2] ESM B sg353[0379][scsi7 port 2]
Enclosure SV13306129 Drawer 1 ESM sg64 11 disks diskset "47887" ESM sg353 11 disks diskset "47887"
Enclosure SV13306129 Drawer 2 ESM sg64 12 disks diskset "53906" ESM sg353 12 disks diskset "53906"
Enclosure SV13306129 Drawer 3 ESM sg64 12 disks diskset "35322" ESM sg353 12 disks diskset "35322"
Enclosure SV13306129 Drawer 4 ESM sg64 12 disks diskset "37055" ESM sg353 12 disks diskset "37055"
Enclosure SV13306129 Drawer 5 ESM sg64 11 disks diskset "16025" ESM sg353 11 disks diskset "16025"
Enclosure SV13306129 sees 58 disks

DCS3700 configuration: 4 enclosures, 1 SSD, 7 empty slots, 233 disks total
Location 0123456789AB-5-12 appears empty but should have an SSD
Location SV12616296-1-3 appears empty but should have an SSD
Location SV12616296-5-12 appears empty but should have an SSD
Location SV11812206-1-3 appears empty but should have an SSD
Location SV11812206-5-12 appears empty but should have an SSD

scsi7[07.00.00.00] 0000:11:00.0 [P2 SV13306129 ESM B (sg353)] [P3 SV12616296 ESM A (sg63)]
[P4 0123456789AB ESM B (sg4)]
scsi8[07.00.00.00] 0000:8b:00.0 [P2 SV13306129 ESM A (sg64)] [P3 SV11812206 ESM B (sg6)]
[P4 0123456789AB ESM A (sg244)]
scsi9[07.00.00.00] 0000:90:00.0 [P3 SV11812206 ESM A (sg66)] [P4 SV12616296 ESM B (sg3)]

##########################################################################
# Beginning enclosure checks.
##########################################################################
Enclosure checks successful.

##########################################################################
# Beginning recovery group checks.
##########################################################################
Found recovery group BB1RGR, primary server is not the active server.

##########################################################################
# Beginning pdisk checks.
##########################################################################
Found recovery group BB1RGL pdisk e4d5s06 has 0 paths.

#####################################
Beginning IBM Power RAID checks.
#####################################
IBM Power RAID Array is running in degraded mode.
Name   PCI/SCSI Location          Description               Status
------ ------------------------  ------------------------- -----------------
       0007:90:00.0/0:           PCI-E SAS RAID Adapter    Operational
       0007:90:00.0/0:0:1:0      Advanced Function Disk    Failed
       0007:90:00.0/0:0:2:0      Advanced Function Disk    Active
sda    0007:90:00.0/0:2:0:0      RAID 10  Disk Array       Degraded
       0007:90:00.0/0:0:0:0      RAID 10  Array Member     Active
       0007:90:00.0/0:0:3:0      RAID 10  Array Member     Failed
       0007:90:00.0/0:0:4:0      Enclosure                 Active
       0007:90:00.0/0:0:6:0      Enclosure                 Active
       0007:90:00.0/0:0:7:0      Enclosure                 Active
```

See the “gnrhealthcheck script” on page 406 for more information.

Chapter 10. Setting up IBM Storage Scale RAID on the Elastic Storage Server

This section includes information about setting IBM Storage Scale RAID up on the Elastic Storage Server.

It provides information about preparing ESS recovery group servers and creating recovery groups on the ESS.

Configuring IBM Storage Scale RAID recovery groups on the ESS: a sample scenario

This topic provides a detailed example of configuring IBM Storage Scale RAID on an ESS building block. The example considers one GL4 building block, which consists of two recovery group servers cabled to four DCS3700 JBOD disk enclosures, and shows how recovery groups are defined on the disk enclosures. Throughout this topic, it might be helpful to have ESS documentation close at hand.

Preparing ESS recovery group servers

This topic includes information about disk enclosure and host bus adapter (HBA) cabling and provides a procedure for verifying that a GL4 building block is configured correctly.

Disk enclosure and HBA cabling

The GL4 DCS3700 JBOD disk enclosures should be cabled to the intended recovery group servers according to the ESS hardware installation instructions. The GL4 building block contains four disk enclosures. Each disk enclosure contains five disk drawers. In each disk enclosure, drawers 2, 3, and 4 contain 12 HDDs. In disk enclosure 1 only, drawers 1 and 5 have one SSD and 11 HDDs. In disk enclosures 2, 3, and 4 (and, in the case of GL6, 5 and 6), drawers 1 and 5 have 11 HDDs and 1 empty slot. In total, a GL4 building block has 4 enclosures, 20 drawers, 232 HDDs, 2 SSDs, and 6 empty slots. (For comparison, a GL6 building block has 6 enclosures, 30 drawers, 348 HDDs, 2 SSDs, and 10 empty slots.) Each disk enclosure provides redundant A and B environmental service modules (ESM) port connections for host server HBA connections. To ensure proper multi-pathing and redundancy, each recovery group server must be connected to ESM A and ESM B using different HBAs. The ESS hardware installation documentation describes precisely which HBA ports must be connected to which disk enclosure ESM ports.

IBM Storage Scale RAID provides system administration tools for verifying the correct connectivity of GL4 and GL6 disk enclosures, which will be seen later during the operating system preparation.

When the ESM ports of the GL4 disk enclosures have been cabled to the appropriate HBAs of the two recovery group servers, the disk enclosures should be powered on and the recovery group servers should be rebooted.

Verifying that the GL4 building block is configured correctly

Preparation then continues at the operating system level on the recovery group servers, which must be installed with the Red Hat® Enterprise Linux distribution that is provided with ESS. After the cabling is done and the servers have been rebooted, it is necessary to perform a thorough discovery of the disk topology on each server.

To proceed, IBM Storage Scale must be installed on the recovery group servers, but these servers do not need to be added to a GPFS cluster yet. See *IBM Storage Scale: Concepts, Planning, and Installation Guide* for information about installing IBM Storage Scale.

IBM Storage Scale RAID provides tools in `/usr/lpp/mmfs/samples/vdisk` (linked for convenience in the regular `/usr/lpp/mmfs/bin directory`) for collecting and collating information on any attached

GL4 disk enclosures and for verifying that the detected topology is correct. The **mmgetpdisktopology** command examines the list of connected devices for the operating system and produces a colon-delimited database with a line for each discovered GL4 physical disk, disk enclosure ESM expander device, and HBA. **mmgetpdisktopology** should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called **topsummary** concisely summarizes the output of the **mmgetpdisktopology** command.

Create a directory in which to work, and then capture the output of the **mmgetpdisktopology** command from each of the two intended recovery group server nodes:

```
# mkdir gl4
# cd gl4
# ssh server1 /usr/lpp/mmfs/bin/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/bin/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes; the following example is for `server1`:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
DCS3700 enclosures found: SV34604344 SV34615757 SV34617244 SV35229088
Enclosure SV35229088 (number 1):
Enclosure SV35229088 ESM A sg419[0396][scsi6 port 2] ESM B sg244[0396][scsi4 port 2]
Enclosure SV35229088 Drawer 1 ESM sg419 12 disks diskset "11082" ESM sg244 12 disks diskset
"11082"
Enclosure SV35229088 Drawer 2 ESM sg419 12 disks diskset "16380" ESM sg244 12 disks diskset
"16380"
Enclosure SV35229088 Drawer 3 ESM sg419 12 disks diskset "11864" ESM sg244 12 disks diskset
"11864"
Enclosure SV35229088 Drawer 4 ESM sg419 12 disks diskset "31717" ESM sg244 12 disks diskset
"31717"
Enclosure SV35229088 Drawer 5 ESM sg419 12 disks diskset "11824" ESM sg244 12 disks diskset
"11824"
Enclosure SV35229088 sees 60 disks
Enclosure SV34604344 (number 2):
Enclosure SV34604344 ESM A sg185[0396][scsi4 port 1] ESM B sg57[0396][scsi2 port 2]
Enclosure SV34604344 Drawer 1 ESM sg185 11 disks diskset "60508" ESM sg57 11 disks diskset
"60508"
Enclosure SV34604344 Drawer 2 ESM sg185 12 disks diskset "29045" ESM sg57 12 disks diskset
"29045"
Enclosure SV34604344 Drawer 3 ESM sg185 12 disks diskset "08276" ESM sg57 12 disks diskset
"08276"
Enclosure SV34604344 Drawer 4 ESM sg185 12 disks diskset "28750" ESM sg57 12 disks diskset
"28750"
Enclosure SV34604344 Drawer 5 ESM sg185 11 disks diskset "34265" ESM sg57 11 disks diskset
"34265"
Enclosure SV34604344 sees 58 disks
Enclosure SV34617244 (number 3):
Enclosure SV34617244 ESM A sg7[0396][scsi2 port 1] ESM B sg365[0396][scsi6 port 1]
Enclosure SV34617244 Drawer 1 ESM sg7 11 disks diskset "36679" ESM sg365 11 disks diskset
"36679"
Enclosure SV34617244 Drawer 2 ESM sg7 12 disks diskset "18808" ESM sg365 12 disks diskset
"18808"
Enclosure SV34617244 Drawer 3 ESM sg7 12 disks diskset "55525" ESM sg365 12 disks diskset
"55525"
Enclosure SV34617244 Drawer 4 ESM sg7 12 disks diskset "51485" ESM sg365 12 disks diskset
"51485"
Enclosure SV34617244 Drawer 5 ESM sg7 11 disks diskset "24274" ESM sg365 11 disks diskset
"24274"
Enclosure SV34617244 sees 58 disks
Enclosure SV34615757 (number 4):
Enclosure SV34615757 ESM A sg305[0396][scsi5 port 2] ESM B sg125[0396][scsi3 port 2]
Enclosure SV34615757 Drawer 1 ESM sg305 11 disks diskset "59007" ESM sg125 11 disks diskset
"59007"
Enclosure SV34615757 Drawer 2 ESM sg305 12 disks diskset "01848" ESM sg125 12 disks diskset
"01848"
Enclosure SV34615757 Drawer 3 ESM sg305 12 disks diskset "49359" ESM sg125 12 disks diskset
"49359"
Enclosure SV34615757 Drawer 4 ESM sg305 12 disks diskset "62742" ESM sg125 12 disks diskset
"62742"
Enclosure SV34615757 Drawer 5 ESM sg305 11 disks diskset "62485" ESM sg125 11 disks diskset
"62485"
Enclosure SV34615757 sees 58 disks

GSS configuration: 4 enclosures, 2 SSDs, 6 empty slots, 234 disks total, 6 NVRAM partitions
```

```
scsi3[19.00.00.00] U78CB.001.WZS00KG-P1-C2-T1  [P2 SV34615757 ESM B (sg125)]
scsi4[19.00.00.00] U78CB.001.WZS00KG-P1-C2-T2  [P1 SV34604344 ESM A (sg185)] [P2 SV35229088 ESM
B (sg244)]
scsi5[19.00.00.00] U78CB.001.WZS00KG-P1-C3-T1  [P2 SV34615757 ESM A (sg305)]
scsi6[19.00.00.00] U78CB.001.WZS00KG-P1-C3-T2  [P1 SV34617244 ESM B (sg365)] [P2 SV35229088 ESM
A (sg419)]
scsi0[19.00.00.00] U78CB.001.WZS00KG-P1-C11-T1
scsi2[19.00.00.00] U78CB.001.WZS00KG-P1-C11-T2 [P1 SV34617244 ESM A (sg7)]   [P2 SV34604344 ESM
B (sg57)]
```

This output shows a correctly cabled and populated GL4 installation. Four disk enclosures with serial numbers SV34615757, SV34617244, SV35229088, and SV34604344 are found cabled in the correct order and with the correct numbers of disks.

The section for each enclosure begins with the enclosure number as determined by the cabling, followed by which ESM and HBA ports provide connectivity to the enclosure; the following line indicates that `/dev/sg305` is the SCSI/SES expander device representing the enclosure's A ESM, which is at firmware level 0396 and is connected to port 2 of the HBA that the operating system configured as SCSI host 5:

```
Enclosure SV34615757 ESM A sg305[0396][scsi5 port 2] ESM B sg125[0396][scsi3 port 2]
```

Each drawer of an enclosure is represented by a line indicating how many disks are seen over each ESM and a "diskset" checksum of the disk World Wide Names (WWNs), which is used to verify that each ESM connection sees the same set of disks over different device paths.

If the cabling were incorrect in any way, one or more of the enclosures would indicate "number undetermined", and the following message would be printed:

```
Unable to determine enclosure order; check the HBA to enclosure cabling.
```

If any slots are unexpectedly empty or contain the wrong type of disk, the **topsummary** output will include additional messages such as these:

```
Location SV34615757-2-10 appears only on the sg305 path
Location SV34617244-1-3 appears empty but should have an SSD
Location SV34615757-5-12 has an HDD but should be empty
```

The HBAs are also listed (firmware levels in brackets) with their bus addresses and port connections. For example, `[P2 SV34615757 ESM B (sg125)]` indicates that HBA port 2 is connected to ESM B of enclosure SV34615757; ESM B is represented by the `/dev/sg125` SCSI/SES expander device. The HBA location codes and port connections are standard in an ESS building block and are used by **topsummary** to validate the correctness of the cabling.

If the cabling is called out as incorrect and the enclosure order cannot be determined, or if other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), corrections should be made and verified before proceeding. This would probably involve shutting down the servers and the disk enclosures and fixing the cabling and disk placement.

The ESS configuration line for the server topology should indicate the presence of six NVRAM partitions. These are not in the disk enclosures, but are local to the server. Two of them will be used to provide high-speed update logging for IBM Storage Scale RAID (the other four are reserved for possible later use).

The `server2.top` topology file should also be examined using the **topsummary** sample script and verified to be correct. It should also be verified to have found the same enclosures with the same serial numbers and in the same order. One way to do this is to run the following commands and verify that they give exactly the same output:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
# /usr/lpp/mmfs/samples/vdisk/topsummary server2.top | grep number
```

After the GL4 disk enclosure topologies are verified to be correct on both intended recovery group server nodes, it is highly recommended that the disk enclosures be entered into the IBM Storage Scale RAID component database. This allows the system administrator to provide a meaningful name for each component and to record each of their physical rack locations in a manner intended to make maintenance actions easier.

To proceed (either to populate the IBM Storage Scale RAID component database or to create recovery groups), the recovery group servers must now be members of the same GPFS cluster. See the *IBM Storage Scale: Administration Guide* for information about creating a GPFS cluster.

After the intended recovery group servers are added to a GPFS cluster, the component database can be updated to identify the equipment rack and the U compartments in which the disk enclosures reside.

In this scenario, a GL4 building block will occupy a 42U Primary Rack. Using the component database, the four disk enclosures and the rack in which they reside may be given meaningful names and associated with each other.

To create the component database entry for a 42U Primary Rack, which has part number 1410HPA, and to give it the descriptive name BB1, use the **mmaddcomp** command:

```
# mmaddcomp 1410HPA --name BB1
```

Do the same for each of the four disk enclosures as identified by the **topsummary** command. In this example, the enclosures were identified according to the cabling in this order:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
Enclosure SV35229088 (number 1):
Enclosure SV34604344 (number 2):
Enclosure SV34617244 (number 3):
Enclosure SV34615757 (number 4):
```

Suppose this is GL4 building block 1, and the desired descriptive names for the disk enclosures are BB1ENC1, BB1ENC2, BB1ENC3, and BB1ENC4. To define these to the component database, use the disk enclosure part number 1818-80E as follows:

```
# mmaddcomp 1818-80E --serial-number SV35229088 --name BB1ENC1
# mmaddcomp 1818-80E --serial-number SV34604344 --name BB1ENC2
# mmaddcomp 1818-80E --serial-number SV34617244 --name BB1ENC3
# mmaddcomp 1818-80E --serial-number SV34615757 --name BB1ENC4
```

You can also use the **mmdiscovercomp** command in place of **mmaddcomp**, then use **mmchcomp** to assign descriptive names to the enclosures.

At this point, the building block rack and disk enclosures can be listed from the component database using the **mmlscomp** command:

```
# mmlscomp
Rack Components
Comp ID Part Number Serial Number Name
------- ----------- ------------- ------
      1 1410HPA                   BB1

Storage Enclosure Components

Comp ID Part Number Serial Number Name    Display ID
------- ----------- ------------- ------- ----------
      2 1818-80E    SV35229088    BB1ENC1
      3 1818-80E    SV34604344    BB1ENC2
      4 1818-80E    SV34617244    BB1ENC3
      5 1818-80E    SV34615757    BB1ENC4
```

Each of the defined components has an ID. The location of the enclosures can be defined according to the four U compartments they occupy in the rack. To do this, use the **mmchcomploc** command to specify the

component IDs of the enclosures, the rack that contains them, and the starting U compartment (position) within the rack. The syntax of the **mmchcomploc** command is:

```
mmchcomploc Component Container Position
```

To define enclosure BB1ENC1 as occupying U compartments 1 through 4 in rack BB1, use this command:

```
# mmchcomploc BB1ENC1 BB1 1
```

Suppose BB1ENC2, BB1ENC3, and BB1ENC4 have starting U compartments 5, 13, and 17, respectively, in rack BB1. Use the following commands to define these relationships:

```
# mmchcomploc BB1ENC2 BB1 5
# mmchcomploc BB1ENC3 BB1 13
# mmchcomploc BB1ENC4 BB1 17
```

The defined component locations can be listed with the **mmlscomploc** command:

```
# mmlscomploc
Component Location
--------- ------------------
BB1ENC1 Rack BB1 U01-04
BB1ENC2 Rack BB1 U05-08
BB1ENC3 Rack BB1 U13-16
BB1ENC4 Rack BB1 U17-20
```

U compartments 9 - 10 and 11 - 12 are presumably occupied by the recovery group servers. They are omitted here as extraneous to this set of examples. They can be named and defined if desired by using component part number 7915AC1 with the **mmaddcomp** command, and then using their component IDs and starting U compartments with the **mmchcomploc** command, as was done with the disk enclosures.

With the component names and locations entered into the IBM Storage Scale RAID component database as shown, a common maintenance action such as replacing a disk will be able to supply meaningful direction to the user in locating the equipment in a crowded machine room.

Creating recovery groups on the ESS

You can create recovery groups using IBM Storage Scale RAID commands. You can create vdisks, NSDs, and file systems using IBM Storage Scale commands or using the ESS GUI.

To create vdisks, NSDs, and file systems using the ESS GUI, use the **Create File System** action in the **Files > File Systems** view. You must decide whether you will use IBM Storage Scale commands or the ESS GUI to create vdisks, NSDs, and file systems. A combination of the two is not supported. The ESS GUI cannot create a file system on existing NSDs.

Configuring GPFS nodes to be recovery group servers

Having verified the disk enclosure connectivity of the GL4 building block, and having optionally also created a component database to associate names and machine room locations with the storage hardware, the recovery groups may now be created on the GL4 building block disk enclosures and servers.

The servers must be members of the same GPFS cluster and must be configured for IBM Storage Scale RAID.

IBM Storage Scale must be running on the servers, and the servers should not have been rebooted or had their disk configuration changed since the verified disk topology files were acquired.

Defining the recovery group layout

The definition of recovery groups on a GL4 building block is accomplished by dividing the drawers of the enclosures into left and right halves. The sharing of GL4 disk enclosures by two servers implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each enclosure and drawer should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks in the left half of each drawer, slots 1 through 6, and one on the disks in the right half of each drawer, slots 7 through 12. The SSD in drawer 1, slot 3 of the first enclosure will make up the SSD declustered array for the left recovery group, and the SSD in drawer 5, slot 12 of the first enclosure will make up the SSD declustered array of the right recovery group. The remaining 116 HDDs in each half are divided into two vdisk data declustered arrays of 58 disks.

IBM Storage Scale RAID provides a script, **mkrginput**, that understands the layout of ESS building blocks and will automatically generate the **mmcrrecoverygroup** stanza files for creating the left and right recovery groups. The **mkrginput** script, when supplied with the output of the **mmgetpdisktopology** command from the two servers, will create recovery group stanza files for the left and right sets of disks.

In the same directory in which the verified correct topology files of the two servers are stored, run the **mkrginput** command on the two topology files:

```
# mkrginput server1.top server2.top
```

(In ESS 3.5, the **-s** parameter can be used with **mkrginput** to create a single data declustered array in GL4 and GL6 building blocks. Do this only if all GL4 and GL6 recovery groups in the cluster are to be treated the same. See the “mkrginput script” on page 409 for more information.)

This will create two files, one for the left set of disks and one for the right set of disks found in the `server1` topology. The files will be named after the serial number of the enclosure determined to be first in the topology, but each will contain disks from all four enclosures. In this case, the resulting stanza files will be `SV35229088L.stanza` for the left half and `SV35229088R.stanza` for the right half:

```
# ls -l SV35229088L.stanza SV35229088R.stanza
-rw-r--r-- 1 root root 7244 Nov 25 09:18 SV35229088L.stanza
-rw-r--r-- 1 root root 7243 Nov 25 09:18 SV35229088R.stanza
```

The recovery group stanza files will follow the recommended best practice for a GL4 building block of defining in each half a declustered array called NVR with two NVRAM partitions (one from each server) for fast recovery group RAID update logging; a declustered array called SSD with either the left or right SSD to act as a backup for RAID update logging; and two file system data declustered arrays called DA1 and DA2 using the regular HDDs. (If the `-s` parameter was used with **mkrginput**, there will be no DA2 data declustered array.)

The declustered array definitions and their required parameters can be seen by grepping for `daName` in the stanza files:

```
# grep daName SV35229088L.stanza
%da: daName=SSD spares=0 replaceThreshold=1 auLogSize=120m
%da: daName=NVR spares=0 replaceThreshold=1 auLogSize=120m nspdEnable=yes
%da: daName=DA1 VCDSpares=31
%da: daName=DA2 VCDSpares=31
```

The parameters after the declustered array names are required exactly as shown. (If the **-s** parameter was used with **mkrginput**, there will be no DA2 data declustered array, and the parameters for the DA1 data declustered array will instead be `spares=4 VCDSpares=60`.)

The disks that have been placed in each declustered array can be seen by grepping for example for `da=NVR` in the stanza file:

```
# grep da=NVR SV35229088L.stanza
```

```
%pdisk: pdiskName=n1s01 device=//server1/dev/sda5 da=NVR rotationRate=NVRAM
%pdisk: pdiskName=n2s01 device=//server2/dev/sda5 da=NVR rotationRate=NVRAM
```

This shows that a pdisk called `n1s01` will be created using the `/dev/sda5` NVRAM partition from `server1`, and a pdisk called `n2s01` will use the `/dev/sda5` NVRAM partition on `server2`. The parameters again are required exactly as shown. The name `n1s01` means node 1, slot 1, referring to the server node and the NVRAM partition, or "slot." Similarly, `n2s01` means server node 2, NVRAM slot 1.

The disks in the SSD, DA1, and DA2 declustered arrays can be found using similar grep invocations on the stanza files.

If, as was recommended, the IBM Storage Scale RAID component database was used to provide meaningful names for the GL4 building block components, the names of the recovery groups should be chosen to follow that convention. In this example, the building block rack was named BB1 and the enclosures were named BB1ENC1, BB1ENC2, BB1ENC3, and BB1ENC4. It would make sense then to name the left and right recovery groups BB1RGL and BB1RGR. Other conventions are possible as well.

The left and right recovery group stanza files can then be supplied to the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup BB1RGL -F SV35229088L.stanza --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV35229088R.stanza --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

The left recovery group is created with `server1` as primary and `server2` as backup, and the right recovery group is created with `server2` as primary and `server1` as backup.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created (BB1RGL shown):

```
# mmlsrecoverygroup BB1RGL -L

                   declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGL                        4       0     119  4.1.0.1

 declustered   needs                              replace                 scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD          no            0       1     0,0          1     186 GiB   14 days  repair-RGD/VCD 10%  low
 NVR          no            0       2     0,0          1    3744 MiB   14 days  repair-RGD/VCD 10%  low
 DA1          no            0      58    2,31          2     138 TiB   14 days  repair-RGD/VCD 10%  low
 DA2          no            0      58    2,31          2     138 TiB   14 days  repair-RGD/VCD 10%  low

                                        declustered                              checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------

 config data         declustered array   VCD spares     actual rebuild spare space         remarks
 ------------------  ------------------  -------------  ---------------------------------  ----------------
 rebuild space       DA1                 31             36 pdisk
 rebuild space       DA2                 31             36 pdisk

 config data     max disk group fault tolerance  actual disk group fault tolerance  remarks
 --------------  ------------------------------ ---------------------------------- -----------
 rg descriptor  1 enclosure + 1 drawer        1 enclosure + 1 drawer        limiting fault tolerance
 system index   2 enclosure                   1 enclosure + 1 drawer        limited by rg descriptor

 active recovery group server                    servers
 ----------------------------------------------  -------
 server1                                         server1,server2
```

Notice that the vdisk information for the newly-created recovery group is indicated with 0s or is missing; the next step is to create the vdisks.

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

The internal RAID transaction and update log vdisks must be created first.

Each of the GL4 left and right recovery groups will now require:

- A log tip vdisk (type `vdiskLogTip`) in the NVR declustered array
- A log tip backup vdisk (type `vdiskLogTipBackup`) in the SSD declustered array
- A log home vdisk (type `vdiskLog`) in the DA1 declustered array
- A log reserved vdisk (type `vdiskLogReserved` in the DA2 declustered array (and in the DA3 declustered array, in the case of GL6)

These all need to be specified in a log vdisk creation stanza file. (If the **-s** parameter was used with **mkrginput**, disregard references to the log reserved vdisk and DA2 in the remainder of this example.)

On Power Systems servers, the **checksumGranularity=4k** parameter is required for the various log vdisks in the log vdisk stanza file. This parameter should be omitted on non-Power servers.

The log vdisk creation file for a GL4 building block with NVRAM partitions will look like this:

```
# cat mmcrvdisklog.BB1
%vdisk:
  vdiskName=BB1RGLLOGTIP
  rg=BB1RGL
  daName=NVR
  blocksize=2m
  size=48m
  raidCode=2WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogTip

%vdisk:
  vdiskName=BB1RGLLOGTIPBACKUP
  rg=BB1RGL
  daName=SSD
  blocksize=2m
  size=48m
  raidCode=Unreplicated
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogTipBackup

%vdisk:
  vdiskName=BB1RGLLOGHOME
  rg=BB1RGL
  daName=DA1
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLog
  longTermEventLogSize=4m
  shortTermEventLogSize=4m
  fastWriteLogPct=90

%vdisk:
  vdiskName=BB1RGLDA2RESERVED
  rg=BB1RGL
  daName=DA2
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogReserved

%vdisk:
  vdiskName=BB1RGRLOGTIP
  rg=BB1RGR
  daName=NVR
  blocksize=2m
  size=48m
```

```
  raidCode=2WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogTip

%vdisk:
  vdiskName=BB1RGRLOGTIPBACKUP
  rg=BB1RGR
  daName=SSD
  blocksize=2m
  size=48m
  raidCode=3WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogTipBackup

%vdisk:
  vdiskName=BB1RGRLOGHOME
  rg=BB1RGR
  daName=DA1
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLog
  longTermEventLogSize=4m
  shortTermEventLogSize=4m
  fastWriteLogPct=90

%vdisk:
  vdiskName=BB1RGRDA2RESERVED
  rg=BB1RGR
  daName=DA2
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  checksumGranularity=4k        # Power only
  diskUsage=vdiskLogReserved
```

The parameters chosen for size, blocksize, raidCode, fastWriteLogPct, and the event log sizes are standard and have been carefully calculated, and they should not be changed. The only difference in the vdisk log stanza files between two building blocks will be in the recovery group and vdisk names. (In the case of a GL6 building block with NVRAM partitions, there will be an additional vdiskLogReserved for DA3, with parameters otherwise identical to the DA2 log reserved vdisk.)

The **checksumGranularity=4k** parameter is required for Power Systems servers. It should be omitted on non-Power servers.

The log vdisks for the sample GL4 building block BB1 can now be created using the **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisklog.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGLDA2RESERVED
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGRDA2RESERVED
mmcrvdisk: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

You can use the **mmlsvdisk** command (or the **mmlsrecoverygroup** command) to verify that the log vdisks have been created:

```
# mmlsvdisk
                                                      declustered  block size
 vdisk name          RAID code        recovery group     array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 BB1RGLDA2RESERVED   4WayReplication  BB1RGL              DA2                2048  logRsvd
 BB1RGLLOGHOME       4WayReplication  BB1RGL              DA1                2048  log
 BB1RGLLOGTIP        2WayReplication  BB1RGL              NVR                2048  logTip
 BB1RGLLOGTIPBACKUP  Unreplicated     BB1RGL              SSD                2048  logTipBackup
 BB1RGRDA2RESERVED   4WayReplication  BB1RGR              DA2                2048  logRsvd
 BB1RGRLOGHOME       4WayReplication  BB1RGR              DA1                2048  log
```

```
  BB1RGRLOGTIP        2WayReplication  BB1RGR              NVR                2048  logTip
  BB1RGRLOGTIPBACKUP  Unreplicated     BB1RGR              SSD                2048  logTipBackup
```

Now the file system vdisks may be created.

Data vdisks are required to be defined in the two data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics:

- one using 4-way replication and a 1 MiB block size and a total vdisk size of 2048 GiB suitable for file system metadata
- one using Reed-Solomon 8 + 3p encoding and an 16 MiB block size suitable for file system data

The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must of course be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template. The sample vdisk stanza file that is supplied in `/usr/lpp/mmfs/samples/vdisk/vdisk.stanza` can be used for this purpose and adapted to specific file system requirements.

The file system NSD vdisk stanza file in this example looks like this:

```
# cat mmcrvdisknsd.BB1
%vdisk: vdiskName=BB1RGLMETA1
  rg=BB1RGL
  da=DA1
  blocksize=1m
  size=2048g
  raidCode=4WayReplication
  diskUsage=metadataOnly
  failureGroup=1
  pool=system

%vdisk: vdiskName=BB1RGLMETA2
  rg=BB1RGL
  da=DA2
  blocksize=1m
  size=2048g
  raidCode=4WayReplication
  diskUsage=metadataOnly
  failureGroup=1
  pool=system

%vdisk: vdiskName=BB1RGRMETA1
  rg=BB1RGR
  da=DA1
  blocksize=1m
  size=2048g
  raidCode=4WayReplication
  diskUsage=metadataOnly
  failureGroup=1
  pool=system

%vdisk: vdiskName=BB1RGRMETA2
  rg=BB1RGR
  da=DA2
  blocksize=1m
  size=2048g
  raidCode=4WayReplication
  diskUsage=metadataOnly
  failureGroup=1
  pool=system

%vdisk: vdiskName=BB1RGLDATA1
  rg=BB1RGL
  da=DA1
  blocksize=16m
  raidCode=8+3p
  diskUsage=dataOnly
  failureGroup=1
  pool=data
```

```
%vdisk: vdiskName=BB1RGLDATA2
  rg=BB1RGL
  da=DA2
  blocksize=16m
  raidCode=8+3p
  diskUsage=dataOnly
  failureGroup=1
  pool=data

%vdisk: vdiskName=BB1RGRDATA1
  rg=BB1RGR
  da=DA1
  blocksize=16m
  raidCode=8+3p
  diskUsage=dataOnly
  failureGroup=1
  pool=data

%vdisk: vdiskName=BB1RGRDATA2
  rg=BB1RGR
  da=DA2
  blocksize=16m
  raidCode=8+3p
  diskUsage=dataOnly
  failureGroup=1
  pool=data
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as metadataOnly and for placement in the system storage pool, and the file system data vdisks are flagged for eventual dataOnly usage in the data storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see “Creating the GPFS file system” on page 126.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual failureGroup=1 value for the NSDs on the file system vdisks is the same for vdisks in both the BB1RGL and BB1RGR recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the four GL4 disk enclosures, and IBM Storage Scale should be informed of this through a distinct failure group designation for each disk enclosure. It is up to the IBM Storage Scale system administrator to decide upon the failure group numbers for each ESS building block in the GPFS cluster. In this example, the failure group number 1 has been chosen to match the example building block number.

To create the file system NSD vdisks specified in the `mmcrvdisknsd.BB1` file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisknsd.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLMETA1
mmcrvdisk: [I] Processing vdisk BB1RGLMETA2
mmcrvdisk: [I] Processing vdisk BB1RGRMETA1
mmcrvdisk: [I] Processing vdisk BB1RGRMETA2
mmcrvdisk: [I] Processing vdisk BB1RGLDATA1
mmcrvdisk: [I] Processing vdisk BB1RGLDATA2
mmcrvdisk: [I] Processing vdisk BB1RGRDATA1
mmcrvdisk: [I] Processing vdisk BB1RGRDATA2
mmcrvdisk: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

You can use the **mmlsvdisk** command or the **mmlsrecoverygroup** command to verify that the vdisks have been created.

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which IBM Storage Scale builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisknsd.BB1
mmcrnsd: Processing disk BB1RGLMETA1
mmcrnsd: Processing disk BB1RGLMETA2
mmcrnsd: Processing disk BB1RGRMETA1
mmcrnsd: Processing disk BB1RGRMETA2
mmcrnsd: Processing disk BB1RGLDATA1
mmcrnsd: Processing disk BB1RGLDATA2
mmcrnsd: Processing disk BB1RGRDATA1
mmcrnsd: Processing disk BB1RGRDATA2
mmcrnsd: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

```
# mmcrfs gpfsbb1 -F mmcrvdisknsd.BB1 -B 16m --metadata-block-size 1m -T /gpfsbb1 -n 256
The following disks of gpfsbb1 will be formatted on node server1:
BB1RGLMETA1: size 269213696 KB
BB1RGRMETA1: size 269213696 KB
BB1RGLDATA1: size 8593965056 KB
BB1RGRDATA1: size 8593965056 KB
BB1RGLMETA2: size 269213696 KB
BB1RGRMETA2: size 269213696 KB
BB1RGLDATA2: size 8593965056 KB
BB1RGRDATA2: size 8593965056 KB
Formatting file system ...
Disks up to size 3.3 TB can be added to storage pool system.
Disks up to size 82 TB can be added to storage pool data.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
98 % complete on Tue Nov 25 13:27:00 2014
100 % complete on Tue Nov 25 13:27:00 2014
Formatting Allocation Map for storage pool data
85 % complete on Tue Nov 25 13:27:06 2014
100 % complete on Tue Nov 25 13:27:06 2014
Completed creation of file system /dev/gpfsbb1.
mmcrfs: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

The **-n 256** parameter specifies that the allocation maps should account for 256 nodes mounting the file system. This is an example only and should be adjusted to actual cluster expectations.

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Because a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is very simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system using the **mmchpolicy** command:

```
# mmchpolicy gpfsbb1 policy -I yes
Validated policy 'policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude
Rules,
```

```
 0 List Rules, 0 External Pool/List Rules
Policy 'policy'. installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, built on a GL4 building block using two recovery groups, two recovery group servers, four file system metadata vdisk NSDs and four file system data vdisk NSDs, can now be mounted and placed into service:

```
# mmmount gpfsbb1 -a
```

Chapter 11. IBM Storage Scale RAID management API

With the IBM Storage Scale RAID management API, you can develop scripts to automate labor-intensive cluster management tasks. These APIs provide a useful way to integrate and use the ESS system.

The IBM Storage Scale RAID management API is a REST-style interface for managing ESS cluster resources. It runs on HTTPS and uses JSON syntax to frame data inside HTTP requests and responses.

Note: This section covers only the IBM Storage Scale RAID-specific API commands. The API commands that are available with the IBM Storage Scale system are also available to the ESS users. For more information about the API architecture, and the list of IBM Storage Scale management API commands that are available, see IBM Storage Scale management API.

The following IBM Storage Scale RAID management API commands are currently available:

Diskmgmt/vdiskset/server/list/{nodeClass}: GET

Gets a list of IBM Storage Scale RAID servers and their characteristics.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/{nodeClass}` request gets the list of IBM Storage Scale RAID servers and their characteristics. For more information about the fields in the data structures that are returned, see the “mmvdisk server command” on page 350. .

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/nodeClass
```

where

`list/{nodeClass}`
Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Parameters

Table 19. List of parameters		
Parameter name	**Description and applicable keywords**	**Required/Optional**
nodeClass	The node class or the node name for which details are requested.	Required.

Response data

```
  {
  "status": {
    "code": Status code,
    "message": "Status message"
  },  "stdout": [
     "serverList": [
    {
      "nodeName": "Node name",
      "needsAttention": "string",
      "matchingMetric": "string",
      "diskTopology": "Disk topology",
      "number": Node sequence number
    }
    ],
   ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"stdout":"*CLI messages*"
The CLI messages from stdout.

"serverList"
An array that comprises details on the jobs that were run.

"nodeName": "*Name of node*"
The name of the node name for which the details are displayed.

"needsAttention": "*yes | no* "
Specifies whether running the **mmvdisk** command faces problems from internal commands like **mmgetpdisktopology** or **tspsummary**.

"matchingMetric":"*Metrics* "
The match between current disk topology and the expected designs.

"disktopology":"*Disk topology*"
The IBM Storage Scale RAID disk topology on the recovery group servers.

"nodeNumber":"*Node sequence number*"
The sequence number of the node.

Examples

The following example gets information on the nodeClass *nc1*.

Request data:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic
YWRtaW46YWRtaW4wMDE='
'https://198.51.100.1:443/scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/nc1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400® represents an invalid request and 500 represents internal server error.


```
{
  "stdout": [
  "serverList": [
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece01-hs.gpfs.net",
      "nodeNumber": 1
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece02-hs.gpfs.net",
      "nodeNumber": 2
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece03-hs.gpfs.net",
      "nodeNumber": 3
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece04-hs.gpfs.net",
      "nodeNumber": 4
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece05-hs.gpfs.net",
      "nodeNumber": 5
    }
  ]
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```


Related reference

“mmvdisk server command” on page 350
Manages mmvdisk recovery group servers for IBM Storage Scale RAID.

Encryption/clients: GET

Gets information on all the clients that belongs to a specified RKM server.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/encryption/clients` request gets the details for all key clients that are associated with an RKM server. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/clients
```

where

clients
Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 20. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The name of the server where the client is registered.	Optional
clientName	The name of the registered client.	Optional

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "Job status ",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                 {
                   "clientList": [
                  {
                "label": "Client label"
                "keyServer": "Server name"
                "tenantName": "Tenant name"
                "clientName": "Client name"
                "certificateExpiration": "Certificate expiration date"
                "certificateType": "Certificate type"
                }
                ],
```

```
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"label": "*Client name label*"
The label of the registered client.

"keyServer": "*Server name*"
The name of the server.

"tenantName": "*Tenant name*"
The name of the tenant.

"clientName": "*Client name*"
The name of the registered client.

"certificateExpiration": "*Certificate Expiration date* "
The date and time of certificate expiration.

"certificateType": "*Certificate type*"
The type of certificate.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the key clients that are associated with the RKM server.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/encryption/clients'?keyServer=lodestar1.fyre.ibm.com&clientName=myClient1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000001,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:04:41,521",
            "completed": "2021-06-20 09:04:42,129",
            "runtime": 608,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/clients"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client show  --server 'lodestar1.fyre.ibm.com' "
                ],
                "stdout": [
                 {
                 "clientList": [
                 {
                 "certificateExpiration": "2024-06-19 06:45:13 (+0000)",
                 "certificateType": "system-generated",
                 "clientName": "myclient2",
                 "keyServer": "lodestar1.fyre.ibm.com",
                 "label": "myclient2",
                 "tenantNames": "devG1"
                  },
                  {
                  "certificateExpiration": "2021-09-18 10:54:08 (+0000)",
                  "certificateType": "system-generated",
                  "clientName": "myclient3",
                  "keyServer": "lodestar1.fyre.ibm.com",
                  "label": "myclient3",
                  "tenantNames": "(none)"
                  }
                  ],
                 "stderr": [],
                 "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/keys: GET

Displays information about the encryption keys in a tenant.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/encryption/keys` request displays information about the encryption keys in a tenant. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/keys
```

where

keys
Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 21. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the parameters that need to be passed on to the IBM Storage Scale Erasure Code Edition system to complete the requested operation.	Required

Request data

The following list of attributes is available in the request data:

```
{
  "serverName": "string",
  "tenantName": "string",
  "passwordFile": "string"
}
```

The details of the parameters are given in the following list:

"serverName": *Server name*
Specifies the host name or IP address of the RKM server to which the tenant belongs.

"tenantName": "*Tenant name*
Specifies the name of the tenant that comprises the encryption keys.

"passwordFile": "*Password file name*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"info":"*Key details*"
The encryption key details.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the encryption key.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'  -d '{ \
   "serverName": "lodestar1.fyre.ibm.com",
  "tenantName": "devG1",
  "passwordFile": "/var/lib/mmfs/gui/passfile" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/keys'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000001,
            "status": "COMPLETED",
            "submitted": "2021-06-18 16:50:17,335",
            "completed": "2021-06-18 16:50:19,999",
            "runtime": 2664,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/keys"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv key show --server 'lodestar1.fyre.ibm.com' --server-pwd '/root/
passfile1' --tenant 'devG1' "
                ],
                "stdout": [
                    "KEY-fa079f5-9e276a98-4d51-4df0-aeff-f296a3cbd7d2",
                    "info: KEY-fa079f5-9e276a98-4d51-4df0-aeff-f296a3cbd7d2\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
```

```
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/servers: GET

Displays information about RKM servers.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/encryption/servers` request displays information about remote key manager (RKM) servers. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/servers
```

where

servers
: Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 22. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The hostname or IP address of the RKM server for which the details are displayed.	Required

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
```

```
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                 "type": "Server type"
                 "backupKeyServers": "Server type"
                 "distribute": "Keystore file"
                 "fips1402mode": "Password"
                 "interval": "Certification file label"
                 "ipa": "Tenant name"
                 "keyServer": "Tenant name"
                 "kmipCertificateExpiration": "Tenant name"
                 "label": "Tenant name"
                 "nistCompliance": "Tenant name"
                 "restCertificateExpiration": "Tenant name"
                 "restPort": "Port number"
                 "retry": "Number of attempts"
                 "timeout": "Number of attempts"
                 "userID": "Number of attempts"
                 ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
        }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"backupKeyServers": "*Back up servers*
The comma-separated list of server names that is added to the list of backup RKM servers in the RKM.conf file.

"distribute": "*yes | no*
Specifies whether the list of RKM server names (main RKM server and backup RKM servers) in the RKM.conf file are arranged in a different order on each node so that each node connects with the servers in a different order.

"fips1402mode": "*on | off*
Specifies whether the FIPS 140-2 certified encryption model is enabled for communications between the nodes.

"interval": "*Time*
The number of microseconds to wait between connection retries. The valid range is 1 - 1000000000. The default value is 10000 (0.1 seconds).

"ipa": "*IP address*
The server IP address.

"keyServer": "*Server name*
The RKM server name.

"kmipCertificateExpiration": "*Date and time*
The expiration date and time of the non-self-signed certificate files in a certificate chain that is used by the specified key server to establish communication on the KMIP port .

"label": "*Server label*
The label to identify the RKM server.

"nistCompliance": "*off |SP800-131A*
Specifies whether GPFS operates in the NIST 800-131A mode.

"restCertificateExpiration": "*Time and date*
The expiration date and time of the non-self-signed certificate files in a certificate chain that is used by the specified key server to establish communication on the REST port.

"restPort": "*REST Port number*
The port number for the Representational State Transfer (REST) interface.

"retry": "*Number of attempts*
The number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries. The default value is three retries.

"timeout": "*Time*
The connection timeout, in seconds, for retrieving an MEK from an RKM server. The valid range is 1 - 120 seconds. The default value is 60 seconds.

"type": "*Server type*
The type of server.

"userID": "*REST user ID*"
The user ID for the RKM server. The default value is SKLMAdmin.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the RKM server.

Request data:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: text/html' 'https://
198.51.100.1:443/scalemgmt/v2/encryption/servers?keyServer=lodestar1.fyre.ibm.com'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
    "jobs": [
        {
            "jobId": 3000000000011,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:22:28,005",
            "completed": "2021-06-21 05:22:28,309",
            "runtime": 304,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/servers"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv rkm show "
                ],
                "stdout": [
                {
                "serverList": [
                {
                "backupKeyServers": "",
                "distribute": "yes",
                "fips1402mode": "off",
                "interval": 10000,
                "ipa": "9.30.252.171",
                "keyServer": "lodestar1.fyre.ibm.com",
                "kmipCertificateExpiration": "2024-05-23 11:53:14 (+0000)",
                "label": "1_lodestar1",
                "nistCompliance": "on",
                "restCertificateExpiration": "2022-05-23 11:01:35 (+0000)",
                "restPort": "9443",
                "retry": 3,
                "timeout": 60,
                "type": "ISKLM",
               "userID": "SKLMAdmin"
               }
               ],
               "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```


Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/rkms: GET

Provides details about an RKM stanza.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkms` request displays information on a remote key manager (RKM) server stanza. The RKM stanza refers to the block of configuration information that describes a registered key client. It is stored in the `RKM.conf` file. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkms
```

where

rkms
: Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                 "type": "Server type"
                 "kmipServerUri": "Server type"
                 "keyStore": "Keystore file"
                 "passphrase": "Password"
                 "clientCertLabel": "Certification file label"
                 "tenantName": "Tenant name
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
```

```
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"type": "*Server type*
The type of server.

"kmipServerUri": "*Server location*
The Server URL.

"keyStore": "*Keystore file*"
The keystore file.

"passphrase"
The Keystore password.

"clientCertLabel": "*Certification file label.*
Client certification file name.

"tenantName": "*Tenant name*"
The name of the tenant.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the RKM server stanza.

Request data:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic
YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/encryption/rkms'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 3000000000011,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:22:28,005",
            "completed": "2021-06-21 05:22:28,309",
            "runtime": 304,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkms"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv rkm show "
                ],
                "stdout": [
                    "lodestar1_devG1 {",
                    "  type = ISKLM",
                    "  kmipServerUri = tls://9.30.252.171:5696",
                    "  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_lodestar1.myclient2.1.p12",
                    "  passphrase = Ar1cent@123456Nigam",
                    "  clientCertLabel = myclient2",
                    "  tenantName = devG1",
                    "}",
                    "info: lodestar1_devG1 {\n  type = ISKLM\n  kmipServerUri = tls://
9.30.252.171:5696\n  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_lodestar1.myclient2.1.p12\n
passphrase = Ar1cent@123456Nigam\n  clientCertLabel = myclient2\n  tenantName = devG1\n}\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/tenants: GET

Displays information about tenants and RKM servers.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET scalemgmt/v2/encryption/tenants` request displays information about tenants and remote key manager (RKM) servers that they are associated with. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/tenants
```

where

tenants
Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 23. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The name of the server where the tenant exists.	Optional
tenantName	The name of the tenant for which details are requested.	Optional

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                "keyServer": Key server name"
                "registeredClient": Client details
                "rkmId": "Server ID"
                "tenantName" : "Tenant name"
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
```

```
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"keyServer": *Server name*"
The server details.

"registeredClient": *Client details*
The client registered with the server.

"rkmId": "*Server ID*"
The ID of the RKM server.

"tenantName": "*Tenant name*"
The name of the tenant.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the tenant *devg1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json' 'https://198.51.100.1:443//
scalemgmt/v2/encryption/tenants?keyServer=lodestar1.fyre.ibm.com&tenantName=devG1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000004,
            "status": "COMPLETED",
            "submitted": "2021-06-18 14:36:37,173",
            "completed": "2021-06-18 14:36:37,377",
            "runtime": 204,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/tenants"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv tenant show  --server 'lodestar1.fyre.ibm.com' "
                ],
                "stdout": [
                    "devG1",
                    "\tKey Server:          lodestar1.fyre.ibm.com",
                    "\tRegistered Client:   (none)",
                    "\tRKM Id:              (none)",
                    "info: devG1\n\tKey Server:lodestar1.fyre.ibm.com\n\tRegistered
                    Client:(none)\n\tRKM Id: (none)\n\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Recoverygroups/declusteredArray: GET

Gets information on a list of declustered array for all the recovery groups.

Availability

Available on Elastic Storage Server and IBM Storage Scale Erasure Code editions.

Description

The `GET scalemgmt/v2/gnr/recoverygroups/declusteredArray` request gets information on a list of declustered array for all the recovery groups in the cluster. For more information about the fields in the data structures that are returned, see the “mmvdisk recoverygroup command” on page 357.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr//recoverygroups/declusteredArray
```

where

recoverygroups/declusteredArray
Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
  {
  "status": {
    "code": Status code,
    "message": "Status message"
  },  "stdout": [
     "declusteredArray": [
    {
      "active": "yes |no ",
      "bgTask": "string",
      "bgTaskPctComplete": "string",
      "daName": "Declustered Array name",
      "freeSpace": Free space available
      "hwType": "Hardware type"
      "mmvdiskControlled": "yes | no"
      "needsService": "yes | no"
      "numPdisk": Number of Pdisks
      "numSpares": Number of spare disks
      "pctFree": Integer
      "rgName": "Recovery group name"
      "rgType": "recovery group type"
      "totalSpace": "Total space available"
      "trimDevice": "yes | no"
    }
    ],
   ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"stdout":"*CLI messages*"
The CLI messages from stdout.

"declusteredArray"
An array that comprises details on the jobs that were run.

"active": *yes | no*
Specifies whether the declustered array is in an active state.

"bgTask": *Task name*
The background task that is running in the declustered array.

"bgTaskPctComplete": ***Integer***
The percentage of completion of the background task.

"daName": ***Declustered array name***
The name of the declustered array.

"freeSpace": ***Space available***
The amount of free space available.

"hwType": ***Hardware type***
The supported hardware type.

"mmvdiskControlled": ***yes | no***
Specifies whether the declustered array belongs to an mmvdisk recovery group.

"needsService": ***yes | no***
Specifies whether the declustered array requires maintenance services.

"numPdisk": ***Integer***
The number of physical disks (disk) available in the declustered array.

"numSpares": ***Integer***
The spare disk space that is reserved for rebuilding the data.

"pctFree": ***Integer***
The percentage of space that is remaining in the declustered array after the vdisk is created.

"rgName": ***Recovery group name***
The name of the recovery group to which the declustered array belongs.

"rgType": ***Recovery group type***
The type of the recovery group to which the declustered array belongs.

"total space": ***Total space available***
The total space available in the declustered array.

"trimDevice": ***yes | no***
Specifies whether the trim function is enabled for the declustered array.

Examples

The following example gets information on the declustered array *DA1*.

Request data:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic
YWRtaW46YWRtaW4wMDE='
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/declusteredArray
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "stdout": {
  "declusteredArrayList": [
    {
      "active": "yes",
      "bgTask": "scrub",
      "bgTaskPctComplete": 47,
      "daName": "DA1",
      "freeSpace": 0,
      "hwType": "HDD",
      "mmvdiskControlled": "yes",
      "needsService": "yes",
      "numPdisk": 0,
      "numSpares": 2,
      "pctFree": 32,
      "rgName": "rg1",
      "rgType": "scale-out",
```

```
      "totalSpace": 0,
      "trimDevice": "no"
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related reference

“mmvdisk recoverygroup command” on page 357
Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Recoverygroups/{rgName}/{oldNode}/replace/{newNode}: PUT

Replaces a server in an mmvdisk scale-out recovery group.

Availability

Available on Elastic Storage Server and IBM Storage Scale Erasure Code editions.

Description

The PUT `scalemgmt/v2/gnr/recoverygroups/{rgName}/{oldNode}/replace/{newNode}` request replaces a failed or unwanted server in a scale-out recovery group that is managed by the **mmvdisk** command. When replaced, the RAID data from the old server's pdisks are migrated to the replacement server's disks. For more information about the fields in the data structures that are returned, see the “mmvdisk recoverygroup command” on page 357.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/gnr/recoverygroups/{rgName}/
{oldNode}/replace/{newNode}
```

where

{oldNode}
: Specifies the resource that must be updated.

{rgName}
: Specifies the recovery group where the node belongs.

{newNode}
: Specifies the new node name.

Request headers

```
Accept: application/json
```

Parameters

Table 24. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
rgName	The recovery group name.	Required
oldNode	The old node name that is replaced.	Required
newNode	The new node name that replaces the old node.	Required

Request data

No Request data.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
         "progress": [],
        "commands": [
          ""
        ],
        "stdout": [
          " "
        ],
        "stderr": [],
        "exitCode": 0
      },
        "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": ***list***
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": ***ReturnCode***
The return code.

Examples

The following example shows how to replace a server in an mmvdisk scale-out recovery group.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/
json' --header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/recoverygroups/rg1/ess-12/replace/ess-11''
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000003,
      "status": "COMPLETED",
      "submitted": "2021-06-18 08:14:31,854",
      "completed": "2021-06-18 08:14:40,799",
      "runtime": 8945,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/gnr/recoverygroups/rg1/ess-12/replace/ess-11'"
      },
      "result": {
        "progress": [],
        "commands": [
          "mmvdisk recoverygroup replace  --recovery-group rg1
          -N ess-12 --new-node ess-11 [--match 100]
                              [--fanout 32] [-v yes] "
        ],
        "stdout": [

         "info: "
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related reference

"mmvdisk recoverygroup command" on page 357

Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Encryption/clients/deregister: PUT

Unregisters a key client from a tenant.

Availability

Available on all IBM Storage Scale editions.

Description

The PUT `/scalemgmt/v2/encryption/clients/deregister` request unregisters the key client from a tenant. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation. .

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/clients/deregister
```

where:

Client
Specifies the resource to be modified.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Key client name",
  "tenantName": "Tenant name",
  "passwordFile": "Password file name",

}
```

The details of the parameters are given in the following list:

"clientName": "*Key client name*"
Specifies the key client that must be unregistered.

"tenantName": "*Tenant name*"
Specifies the name of the tenant to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
```

```
            "commands":"Commands issued",
            "progress":"Request progress,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"""",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to unregister `myclient1` from a tenant.

Request data:

```
 curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient1"
   "tenantName": "devG1",
   "passwordFile": "/tmp/password",
     }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/clients/deregister'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
 {
    "jobs": [
        {
            "jobId": 1000000000008,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:33:40,120",
            "completed": "2021-06-20 09:33:45,648",
            "runtime": 5528,
            "request": {
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/clients/deregister"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client deregister 'myclient1' --tenant 'devG1' --server-pwd '/
root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Deleting the following KMIP certificate with label:
1185675729815192855 9_devG1_1624181101",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: mmkeyserv: Deleting the following KMIP certificate with label:
1185675729815192855 9_devG1_1624181101\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/clients: PUT

Replaces a valid or an expired client certificate in a specified key client.

Availability

Available on all IBM Storage Scale editions.

Description

The PUT `scalemgmt/v2/encryption/clients` request replaces a client certificate, which has either expired or is still valid, for a specified key client. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/clients
```

where

clients
: Specifies the target of the PUT request.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes is available in the request data:

```
{
  "clientName": "Client name",
  "newClientname": "New client name",
  "passwordFile": "Password file name",
  "daysToExpiration": Days till expiry,
  "keyStorePwdFile": "Keystore password file name"
  "clientCertFile": "Client certification file",
  "clientPrivateKeyFile": "Client private key file",
  "caCertFilePrefix": "Path and file name of certifcate prefix",
  "caCertChainFile": "CA certificates file
  "forceFlag":  true | false
  }
```

The details of the parameters are given in the following list.

"clientName": "*Client name*"
: Specifies the name of the key client where you want to update the client certificate.

"newClientName": "*New client name*"
: Specifies the new name of the key client. The name must be within 1 - 16 characters in length. It must be unique within the IBM Storage Scale cluster. If you do not provide a value for this parameter, then the certificate is not replaced with the new name.

"serverName": "*Server name*"
: Specifies the name of the RKM server to which the key client belongs.

"passwordFile": "*Password file*"
: The password file that comprises a password for accessing the RKM server.

"daysToExpiration": "*Number of days till expiration*"
: The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile" : "*Keystore password file*"
The password file that contains a client keystore password.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

"forceflag": "*true | false*"
Specifies whether a self-signed client certificate is generated for the key client.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example replaces the client certificate.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
    "clientName": "myclient1", \
   "newClientname": "myclient1", \
   "passwordFile": "/tmp/password", \
   "daysToExpiration": 1095, \
   "keyStorePwdFile": "/tmp/password", \
   "clientCertFile": "/tmp/cert", \
   "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
   "caCertFilePrefix": "/tmp/cert", \
   "caCertChainFile": "/tmp/CA/certfiles.0.cert", \
   "forceFlag": false \
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/clients'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 3000000000012,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:30:06,455",
            "completed": "2021-06-21 05:30:23,457",
            "runtime": 17002,
            "request": {
                "data": {
                    "clientName": "myclient2",
                    "daysToExpiration": 1095,
                    "keyStorePwdFile": "/home/passfile1",
                    "newClientname": "myclient1",
                    "passwordFile": "/home/passfile1"
                    "clientCertFile": "/tmp/cert", \
                    "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
                    "caCertFilePrefix": "/tmp/cert", \
                    "caCertChainFile": "/tmp/CA/certfiles.0.cert", \
                    "forceFlag": false
                },
                "type": "PUT",
                "url": "/scalemgmt/v2/encryption/clients"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client update 'myclient2' --client 'myclient1' --days 1095 --
keystore-pwd '/home/passfile1' --server-pwd '/home/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: [I] Client currently does not have access to the key.  Continue
the registration process ...",
                    "mmkeyserv: Successfully accepted client certificate",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "mmkeyserv: Deleting the following KMIP certificate with label:
9842179411678971055_devG1_1624267300",
```

```
                    "info: mmkeyserv: [I] Client currently does not have access
to the key.  Continue the registration process ...\nmmkeyserv: Successfully accepted
client certificate\nmmkeyserv: Deleting the following KMIP certificate with label:
9842179411678971055_devG1_1624267300\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```


Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/servers: PUT

Updates an RKM server connection to the IBM Storage Scale Erasure Code Edition cluster.

Availability

Available on all IBM Storage Scale editions.

Description

The PUT `scalemgmt/v2/encryption/servers` request updates a remote key manager (RKM) server connection to the IBM Storage Scale cluster. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/servers
```

where

servers
: Specifies the resource that must be updated.

Request headers

```
Accept: application/json
```

Parameters

Table 25. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the required parameters to be passed on to the IBM Storage Scale Erasure Code Edition system to perform the requested operation	Required

Request data

The following list of attributes is available in the request data:

```
{
 "restPortNumber": Port number,
  "restUserID": "User ID ",
  "passwordFile": "Password file name",
  "accept": true | false,
  "certFilePrefix": "Certificate File prefix",
  "serverNamebackup": "Server name list",
  "dis": true | false,
  "nodis": true | false,
  "connectionTimeout": Time,
  "connectionAttempts": Number of attempts,
  "microseconds": Time
  }
```

The details of the parameters are given in the following list.

"serverName": "*Server name*"
: Specifies the hostname or the IP address of the RKM server.

"restPortNumber": "*REST Port Number*"
: Specifies the port number for the Representational State Transfer (REST) interface on the IBM Security Key Lifecycle Manager server.

"restUserID": "*User ID*"
: Specifies the user ID for the RKM server. The default value is SKLMAdmin.

"passwordFile": "*Password file*"
: Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"accept": "*true | false*"
: Specifies whether the command is configured to automatically accept certificates from the RKM server.

"caCertFilePrefix" : "*Path and file name of the Certificate file prefix* "
: The path and file name prefix of non-self-signed certificate files in a certificate chain.

"serverNamebackup" : "*Server name list*"
: Specifies a comma-separated list of server names that you want to add to the list of backup RKM servers defined in the RKM.conf file.

"dis": "*true | false*"
: Specifies whether the list of RKM server names, including the main RKM server and backup RKM servers, are arranged in the RKM.conf file in a different order on each node. This arrangement ensures that each node connects with the servers in a different order.

"nodis": "*true | false*"
: Specifies whether the list of RKM server names is arranged in the RKM.conf file.

"connectionTimeout": "*Time*"
: Specifies the connection timeout, in seconds, for retrieving a master encryption key (MEK) from an RKM server. The valid range is 1 - 120 seconds and the default value is 60 seconds.

"connectionAttempts": "*Number of attempts*"
: Specifies the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries and the default value is three retries.

"microseconds": "*Time*"
: Specifies the number of microseconds of waiting between attempts to connect. The valid range is 1 - 1000000000 and the default value is 10000 (0.1 seconds).

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
         "progress": [],
        "commands": [
          ""
        ],
        "stdout": [
          " "
        ],
        "stderr": [],
        "exitCode": 0
      },
        "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example updates the connection of the *lodestar1.fyre.ibm.com* server with the IBM Storage Scale cluster.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
  "serverName": "lodestar1.fyre.ibm.com",
  "restPortNumber": 44742,
  "restUserID": "admin",
  "passwordFile": "/var/lib/mmfs/gui/passfile",
  "accept": false,
  "certFilePrefix": "CertFilesPrefix.n.cert",
  "serverNamebackup": "string",
  "dis": true,
  "nodis": false,
  "connectionTimeout": 60,
  "connectionAttempts": 3,
  "microseconds": 10000
   }\
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/servers'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000003,
      "status": "COMPLETED",
      "submitted": "2021-06-18 08:14:31,854",
      "completed": "2021-06-18 08:14:40,799",
      "runtime": 8945,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/encryption/servers"
      },
      "result": {
        "progress": [],
        "commands": [
          "mmkeyserv server update 'lodestar1.fyre.ibm.com' --server-pwd '/root/passfile1' --
accept "
        ],
        "stdout": [
          "mmkeyserv: Propagating the cluster configuration data to all",
          "  affected nodes.  This is an asynchronous process.",
          "info: "
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
```

```
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/servers: POST

Adds an RKM server connection to the IBM Storage Scale Erasure Code Edition cluster.

Availability

Available on all IBM Storage Scale editions.

Description

The POST `scalemgmt/v2/encryption/servers` request adds a remote key manager (RKM) server connection to the IBM Storage Scale cluster. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/servers
```

where

servers
: Specifies the resource that must be added.

Request headers

```
Accept: application/json
```

Parameters

Table 26. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the required parameters to be passed on to the IBM Storage Scale Erasure Code Edition system to perform the requested operation.	Required

Request data

The following list of attributes is available in the request data:

```
{
 "restPortNumber": Port number,
  "restUserID": "User ID ",
  "passwordFile": "Password file name",
  "accept": true | false,
  "certFilePrefix": "Certificate File prefix",
```

```
  "serverNamebackup": "Server name list",
  "dis": true | false,
  "nodis": true | false,
  "connectionTimeout": Time,
  "connectionAttempts": Number of attempts,
  "microseconds": Time
  }
```

The details of the parameters are given in the following list.

"serverName": "*Server name*"
Specifies the hostname or the IP address of the RKM server.

"restPortNumber": "*REST Port Number*"
Specifies the port number for the Representational State Transfer (REST) interface on the IBM Security Key Lifecycle Manager server.

"restUserID": "*User ID*"
Specifies the user ID for the RKM server. The default value is SKLMAdmin.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"accept": "*true | false*"
Specifies whether the command is configured to automatically accept certificates from the RKM server.

"caCertFilePrefix" : "*Path and file name of the Certificate file prefix* "
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"serverNamebackup" : "*Server name list*"
Specifies a comma-separated list of server names that you want to add to the list of backup RKM servers defined in the RKM.conf file.

"dis": "*true | false*"
Specifies whether the list of RKM server names, including the main RKM server and backup RKM servers, are arranged in the RKM.conf file in a different order on each node. This arrangement ensures that each node connects with the servers in a different order.

"nodis": "*true | false*"
Specifies whether the list of RKM server names is arranged in the RKM.conf file.

"connectionTimeout": "*Time*"
Specifies the connection timeout, in seconds, for retrieving a master encryption key (MEK) from an RKM server. The valid range is 1 - 120 seconds and the default value is 60 seconds.

"connectionAttempts": "*Number of attempts*"
Specifies the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries and the default value is three retries.

"microseconds": "*Time*"
Specifies the number of microseconds of waiting between attempts to connect. The valid range is 1 - 1000000000 and the default value is 10000 (0.1 seconds).

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
```

```
      "result": {},
         "progress": [],
        "commands": [
          ""
        ],
        "stdout": [
          " "
        ],
        "stderr": [],
        "exitCode": 0
      },
        "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example adds the *lodestar1.fyre.ibm.com* server connection with IBM Storage Scale cluster.

Request data:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
  "serverName": "lodestar1.fyre.ibm.com",
  "restPortNumber": 44742,
  "restUserID": "admin",
  "passwordFile": "/var/lib/mmfs/gui/passfile",
  "accept": false,
  "certFilePrefix": "CertFilesPrefix.n.cert",
  "serverNamebackup": "string",
  "dis": true,
  "nodis": false,
  "connectionTimeout": 60,
  "connectionAttempts": 3,
  "microseconds": 10000
   }\
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/servers'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000003,
      "status": "COMPLETED",
      "submitted": "2021-06-18 08:14:31,854",
      "completed": "2021-06-18 08:14:40,799",
      "runtime": 8945,
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/encryption/servers"
      },
      "result": {
        "progress": [],
        "commands": [
          "mmkeyserv server add 'lodestar1.fyre.ibm.com' --server-pwd '/root/passfile1' --
accept "
        ],
        "stdout": [
          "mmkeyserv: Propagating the cluster configuration data to all",
          "  affected nodes.  This is an asynchronous process.",
          "info: "
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/tenants: POST

Adds a tenant on RKM servers.

Availability

Available on all IBM Storage Scale editions.

Description

The POST `/scalemgmt/v2/encryption/tenants` request adds a tenant on the remote key manager (RKM) server. A tenant is an IBM Security Key Lifecycle Manager device group that comprises the encryption keys for registered key clients. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/tenants
```

where:

tenants
Specifies the resource to be created.

Request headers

```
Accept: application/json
```

Request data

```
{

  "tenantName": "Tenant name",
  "serverName": "Server name"
  "passwordFile": "Password file name",

}
```

The details of the parameters are given in the following list:

"tenantName": "*Tenant name*"
Specifies the name of the tenant that you want to add to the RKM server.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the tenant belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
```

```
            "progress":"Request progress,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to add a tenant.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "tenantName": "devG1",
   "serverName": "sklm11.fyre.ibm.com",
   "passwordFile": "/tmp/password",
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/tenants'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000002,
            "status": "COMPLETED",
            "submitted": "2021-06-18 14:25:03,238",
            "completed": "2021-06-18 14:25:13,280",
            "runtime": 10042,
            "request": {
                "data": {
                    "passwordFile": "/root/passfile1",
                    "serverName": "lodestar1.fyre.ibm.com",
                    "tenantName": "devG1"
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/tenants"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv tenant add 'devG1' --server 'lodestar1.fyre.ibm.com' --server-
pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: "
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/keys: POST

Creates encryption keys within a tenant.

Availability

Available on all IBM Storage Scale editions.

Description

The POST `/scalemgmt/v2/encryption/keys` request creates an encryption key within a tenant. The key IDs are also displayed on the remote key manager (RKM) server console. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/keys
```

where:

keys
Specifies the resource to be created.

Request headers

```
Accept: application/json
```

Request data

```
{
  "tenantName": "Tenant name",
  "serverName": "Server name",
  "passwordFile": "Password file name",
  "numberOfKeys": Number of keys created
}
```

The details of the parameters are given in the following list.

"tenantName": "*Tenant name*"
Specifies the name of the tenant where the encryption key or keys are created.

"serverName": "*Server name*"
Specifies the hostname or the IP address of the RKM server.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"numberOfKeys": "*Number of keys created*"
Specifies the number of key or keys that are created. The default value is 1.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
```

```
        "result":"",
          {
            "commands":"Commands issued",
            "progress":"Request progress,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
  ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to create an encryption key for the tenant devG1.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "tenantName": "devG1",
  "serverName": "sklm11.fyre.ibm.com",
  "passwordFile": "/tmp/password",
  "numberOfKeys": 1 \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/keys'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000004,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/encryption/keys"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ mmkeyserv key create --server 'sklm11.fyre.ibm.com' [--server-pwd '/tmp/
password']
   --tenant 'devG1' [--count '1'] ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/clients: POST

Creates a key client to connect with the remote key management (RKM) server.

Availability

Available on all IBM Storage Scale editions.

Description

The POST `/scalemgmt/v2/encryption/clients` request creates a key client that communicates with the RKM server. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/clients
```

where:

clients
Specifies the resource to be added.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Client name",
  "serverName": "Server name",
  "passwordFile": "Password file path",
  "daysToExpiration": Number of days till expiration,
  "keyStorePwdFile": "Keystore password file",
  "clientCertFile": "Client certification file",
  "clientPrivateKeyFile": "Client private key file",
  "caCertFilePrefix": "Path and file name of certifcate prefix",
  "caCertChainFile": "CA certificates file"
}
```

The details of the parameters are given in the following list.

"clientName": "*Client name*"
Specifies the name of the key client that is created. The name must be within 1 - 16 characters in length. It must be unique within the IBM Storage Scale cluster.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"daysToExpiration": "*Number of days till expiration*"
The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile" : "*Keystore password file*"
The password file that contains a client keystore password.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

Response data

```
{
  "jobs": [
    {
      "jobId": 3000000000003,
      "status": "RUNNING | COMPLETED | FAILED",
      "submitted": "Time and date",
      "completed": "Time and date",
      "runtime": 7,
      "request": {
        "data": {
          "caCertChainFile": "Password file name",
          "caCertFilePrefix": "Prefix file name",
          "clientCertFile": "Certificate file name",
          "clientName": "Client name",
          "clientPrivateKeyFile": "Private key file",
          "daysToExpiration": Number of days,
          "keyStorePwdFile": "Keystore Password file",
          "passwordFile": "Password file name",
          "serverName": "Server name"
        },
        "type": "POST | GET | PUT | DELETE",
        "url": "Request URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": Request Code,
    "message": "Request message"
  }
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientName": "*Client name*"
Specifies the name of the key client that is created. The name must be within 1 - 16 characters in length. It must be unique within the IBM Storage Scale cluster. Required.

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"daysToExpiration": "*Number of days till expiration*"
The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile": "*Keystore password file*"
The password file that contains a client keystore password.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the key client belongs. Required.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to create a key client `myclient1`.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient1", \
   "serverName": "sklm11.fyre.ibm.com", \
   "passwordFile": "/tmp/password", \
```

```
   "daysToExpiration": 1095, \
   "keyStorePwdFile": "/tmp/password", \
   "clientCertFile": "/tmp/cert", \
   "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
   "caCertFilePrefix": "/tmp/cert", \
   "caCertChainFile": "/tmp/CA/certfiles.0.cert" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/clients'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000002,
            "status": "COMPLETED",
            "submitted": "2021-06-20 06:45:11,894",
            "completed": "2021-06-20 06:45:20,141",
            "runtime": 8247,
            "request": {
                "data": {
                    "clientName": "myclient2",
                    "keyStorePwdFile": "/root/passfile1",
                    "passwordFile": "/root/passfile1",
                    "serverName": "lodestar1.fyre.ibm.com"
                    "clientCertFile": "/tmp/cert", \
                    "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
                    "caCertFilePrefix": "/tmp/cert", \
                    "caCertChainFile": "/tmp/CA/certfiles.0.cert" \
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/clients"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client create 'myclient2' --server 'lodestar1.fyre.ibm.com' --
server-pwd '/root/passfile1' --keystore-pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: "
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/clients/register: POST

Registers a key client to a tenant.

Availability

Available on all IBM Storage Scale editions.

Description

The POST `/scalemgmt/v2/encryption/clients/register` request registers a key client to a tenant. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/clients/register
```

where:

Client
Specifies the resource to be created. Required.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Key client name",
  "tenantName": "Tenant name",
  "passwordFile": "Password file name",
  "rkmID": "string"
}
```

The details of the parameters are given in the following list.

"clientName": "*Key client name*"
Specifies the key client that you want to register. The key client name must be within 1 - 16 characters in length. It must be unique within the IBM Storage Scale cluster. Required.

"tenantName": "*Tenant name*"
Specifies the name of the tenant to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for it when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

"rkmID":"*Server ID*"
Specifies a new remote key manager (RKM) ID. An RKM ID must be unique within the cluster. It must be 1 - 21 characters in length and contain only alphanumeric characters or an underscore (_). It must begin with a letter or an underscore. An RKM ID identifies an RKM stanza in the `RKM.conf` file. The stanza contains the information that a node needs to retrieve a master encryption key (MEK) from an RKM.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
             "progress":"Request progress,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"CLI messages",
           },
```

```
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to register `myclient2` to a tenant.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient2"
   "tenantName": "devG1",
   "passwordFile": "/tmp/password",
    "rkmID": "AU8763hgsu09"
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/clients/register'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000007,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:25:00,801",
            "completed": "2021-06-20 09:25:12,142",
            "runtime": 11341,
            "request": {
                "data": {
                    "clientName": "myclient1",
                    "passwordFile": "/root/passfile1",
                    "rkmID": "lodestar1_devG1",
                    "tenantName": "devG1"
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/clients/register"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client register 'myclient1' --rkm-id 'lodestar1_devG1' --tenant
'devG1' --server-pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: [I] Client currently does not have access to the key.  Continue
the registration process ...",
                    "mmkeyserv: Successfully accepted client certificate",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: mmkeyserv: [I] Client currently does not have access to the key.
Continue the registration process ...\nmmkeyserv: Successfully accepted client certificate\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }


}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/clients/{clientName}: DELETE

Deletes the specified client.

Availability

Available on all IBM Storage Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/clients/{clientName}` request deletes the specified key client. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/clients/{clientName}
```

Request headers

```
Accept: application/json
```

Parameters

Table 27. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
clientName	Name of the key client that must be deleted.	Required

Request data

No request data.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
```

```
      "pids": "Integer"
    }
  ]
  }
  }
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT | DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete a key client:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/encryption/clients/myclient1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/clients/myclient1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv client delete 'myclient1' " ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/tenants: DELETE

Deletes a tenant from the RKM server.

Availability

Available on all IBM Storage Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/tenants` request deletes a tenant from the remote key manager (RKM) server. A tenant is an IBM Security Key Lifecycle Manager device group that comprises the encryption keys for registered key clients. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation .

Request URL

```
https://<IP address or hostname of API server>:<port>/scalemgmt/v2/encryption/ tenants
```

Request headers

```
Accept: application/json
```

Request data

```
{
  "tenantName": "Tenant name",
  "serverName": "Server name",
  "passwordFile": "Password file"
}
```

The details of the parameters are given in the following list.

"tenantName": "*Tenant name*"
The name of the tenant that is deleted.

"serverName": "*Server name*"
The hostname or the IP address of the RKM server to which the tenant belongs.

"passwordFile": "*Password file*"
The password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
}    }
  ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *Commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*Exit codes*"
Specifies the exit code of the command. Zero indicates success while any value other than zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*Request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete a tenant:

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json' -d '{ \
  "tenantName": "devG1",
 "serverName": "sklm11.fyre.ibm.com",
 "passwordFile": "/tmp/password",
 'https://198.51.100.1:443/scalemgmt/v2/encryption/tenants'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
```

```
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/tenants"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv key delete --server 'sklm11.fyre.ibm.com' [--server-pwd '/tmp/
password']
   {--all --tenant 'devG1' | --file '1'}" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Encryption/servers/{serverName}: DELETE

Deletes the specified sever.

Availability

Available on all IBM Storage Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/servers/{serverName}` request deletes the specified remote key manager (RKM) server. For more information about the fields in the data structures that are returned, see the **mmkeyserv** command in the IBM Storage Scale documentation.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/encryption/servers/
{serverName}
```

where

{serverName}
: Specifies the RKM server that must be deleted.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
```

```
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
}    }
  ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete the specified RKM server:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json''https://
198.51.100.1:443/scalemgmt/v2/encryption/servers/sklm11.fyre.ibm.com'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/servers/sklm11.fyre.ibm.com"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv server delete sklm11.fyre.ibm.com" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related information

mmkeyserv command in the IBM Storage Scale documentation.

Filesystems/{filesystemName}/file/{path}: GET

Gets the details on a specific file in a file system or the entire file system, as required.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The GET `filesystems/{filesystemName}/file/{path}` request gets the details on a specific file in a file system or the entire file system, based on the requested parameters included in the endpoint.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/filesystems /filesystemName/file /path
```

where

`file/{path}`
Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Parameters

Table 28. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
filesystemName	The name of the file system where the file belongs.	Required.
path	The directory location. The path that is provided is relative to the file system's mount point.	Required.

Response data

```
  {
  "status": {
    "code": Return Code,
    "message": "Return message"
  },
 "result": {
        "progress": [],
        "commands": [
          "String' "
        ],
        "stdout": [
          "info:   Request details"
        ],
        "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "Request URL",
    },
    }
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands issued*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

"info: *Request details*
The details of the file or file system that is requested.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

Examples

The following example gets information on the file system *cacheFS* located in the path *122Fmyfile*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/CacheFS/file/122Fmyfile
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
   "request": {
        "type": "GET",
        "url": "/scalemgmt/v2/filesystems/cacheFS/file/122Fmyfile"
      },
      "result": {
        "progress": [],
        "commands": [
          "stat '/afm/cacheFS/122Fmyfile' "
        ],
        "stdout": [
          "info:   File: /afm/cacheFS/122Fmyfile\n  Size: 1073741824\tBlocks: 2097152    IO
Block: 262144 regular file\nDevice: 2eh/46d\tInode: 6208        Links: 1\nAccess: (0644/-rw-r--
r--)  Uid: (    0/    root)   Gid: (    0/    root)\nAccess: 2021-05-28 06:09:12.596707000
+0000\nModify: 2021-05-28 06:09:12.597668000 +0000\nChange: 2021-05-28 06:09:12.597668000
+0000\n Birth: -\n"
```

```
        ],
}
```

Filesystems/{filesystemName}/file/{path}/fileSize/{size}: POST

Creates a file of specified size in a GPFS file system and assigns a user, a group, or both as its owner.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `filesystems/`*{filesystemName}*`/file/`*{path}*`/fileSize/`*{size}* request creates a file in a GPFS file system and assigns a user, a group, or both as its owner. You can specify the size of the file that you want to create. You can provide both the ID and the name of the user or group in the request parameter. When the file is created the user or group ID, and not the name, is displayed to indicate the file owner in the log that is generated.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/filesystems/filesystemName/
file/path/fileSize/size
```

Request headers

```
Accept: application/json
```

Request data

The following list of attributes is available in the request data:

```
{
  "user": "String",
  "uid": Integer,
  "group": "String",
  "gid": Integer
  "recursive": true | false

  }
```

The details of the parameters are given in the following list:

"user": ***Owner name***
: The name of the user who owns the file.

"uid": ***User ID***
: The unique identifier of the user who owns the file.

"group": ***Owner group***
: The name of the user group that owns the file.

"gid" : ***Group ID***
: The unique identifier of the user group that owns the file.

"permission":"*Access permissions*"
: The number of permissions that are set by using the CLI command **chmod**. If nothing is specified, then no action is allowed.

"recursive":"*true | false*"
: Specifies whether the owner is defined recursively for the entire file system or for the specific directory that is defined in the endpoint path.

Parameters

Table 29. List of parameters		
Parameter name	**Description and applicable keywords**	**Required/Optional**
filesystemName	Name of the GPFS file system where the file is created.	Required
path	The directory location. The path that is provided is relative to the file system's mount point.	Required
size	The size of the new file.	Required
body	The request body.	Required

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Date and Time",
         "completed":"Date and Time",
         "runtime":Duration",
         "status":"Job status",
         "pids":"Process IDs
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and the time at which the job was submitted.

"completed":"*Date and Time*"
Specifies the date and the time at which the job was completed.

"runtime":"*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":" *Process ID*"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to create a new file of size *1G*.

Request data -

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "user": "testuser55", \
   "uid": 1234, \
   "group": "mygroup", \
   "gid": 4711,
   "recursive": false, \
   "permission": 700 \
 }' 'https://9.114.204.107:2008/scalemgmt/v2/filesystems/NewFS/file/12Fmyfile.img/fileSize/1G'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 563,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/NewFS/file/12Fmyfile.img/fileSize/1G"
      "data": " "user": "testuser55",
                "uid": 1234,
                "group": "mygroup",
```

```
                "gid": 4711"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "fallocate -l length filename" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Filesystems/{filesystemName}/file/{path}: DELETE

Deletes a file from the specified GPFS file system.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The `DELETE filesystems/{filesystemName}/file/{path}` request deletes a file from the specified GPFS file system.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/file/{path}
```

where

`file/{path}`
: Specifies the directory location in the GPFS file system from where the file must be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

Table 30. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
filesystemName	Name of the GPFS file system from which the file must be deleted.	Required
path	The directory location. The path that is provided is relative to the file system's mount point.	

Request data

No request data.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
}    }
  ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete a file that exists in the path *2Fmyfile.img* in the file system *gpfs2*:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs2/file/2Fmyfile.img'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/filesystems/gpfs2/file/2Fmyfile.img"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "rm -rf filename" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Filesystems/{fileSystemName}/vdiskset/{vdiskset}: POST

Adds a IBM Storage Scale RAID vdisk set to mmvdisk file systems.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `filessystems/{filesystemName}/vdiskset/{vdiskSet}` request adds a IBM Storage Scale RAID vdisk set to mmvdisk file systems. For more information about the fields in the data structures that are returned, see the "mmvdisk command" on page 341.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/filesystems/fileSystemName/
vdiskset/vdiskset
```

where:

fileSystemName
Specifies the mmvdisk file system where the vdisk set is to be added. Required.

vdiskset
Specifies the vdisk set that is to be added. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Parameters

Table 31. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
fileSystemName	The name of the file system name where the vdisk is to be added.	Required.
vdiskset	The name of the vdisk that is added.	Minimum 1 Required.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
             "progress":"Request progress,
             "exitCode":"Exit code",
             "stderr":"Error",
```

```
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
  ],
 }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data. Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to add the vdisk set `vs1` to the file system `udata`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/filesystem/udata/vdisk/vs1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000004,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/filesystem/udata/vdisk/vs1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ mmvdisk filesystem add --file-system udata --vdisk-set VS1 ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

"mmvdisk command" on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/diskmgmt/vdiskset: GET

Gets the list of vdisk sets and displays their attributes and sizing information.

Availability

Available on IBM Storage Scale Erasure Code Edition only.

Description

The `GET scalemgmt/v2/gnr/diskmgmt/vdiskset` request displays the list of vdisk sets along with their attributes and sizing information. For more information about the fields in the data structures that are returned, see the "mmvdisk vdiskset command" on page 380.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/diskmgmt/vdiskset
```

where

vdiskset
Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 32. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
vdisksetName	The name of the vdisk set for which the attributes and sizing information is listed.	Optional

Request data

No request data.

Response data

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "vdiskList": [
    {
      "vdisksetName": "string",
      "created": "string",
      "fsName": "string",
      "recoveryGroups": "string",
      "vdiskCount": 0,
      "vdiskSize": 0,
      "vdiskRawSize": 0,
      "declusteredArray": "string",
      "raidCode": "string",
      "blocksize": 0,
      "usage": "string",
      "storagePool": "string",
      "totalRawSize": 0,
      "freeRawSize": 0,
      "vdiskSets": "string",
      "freePercent": 0,
      "hardwareType": "string",
      "memoryAvailable": 0,
      "memoryRequired": "string",
      "vdiskSetServerMemory": "string",
      "nodeClass": "string"
    }
  ]
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"vdiskList"

"vdisksetName": "*Name*"
The vdisk set name.

"blocksize": *Size*
The block size of a vdisk set definition. The block size is constrained by the RAID code. Valid values for 3WayReplication and 4WayReplication are `256K, 512K, 1M, and 2M`. Valid values for 4+2P and 4+3P are `512K, 1M, 2M, 4M, 8M`. Valid values for 8+2P and 8+3P are `512K, 1M, 2M, 4M, 8M, 16M`.

"created": "*yes | no*"
Specifies whether the vdisk set is created.

"declusteredArray": "*name* "
The name of the declustered array.

"freePercent": "*size in Percentage* "
The available aggregate size of the declustered array to which the vdisk set belongs, specified as a percentage.

"freeRawSize": "*size in bytes* "
The available aggregate size of the declustered array to which the vdisk set belongs, specified as bytes.

"fsName": "*name* "
The name of the file system that uses this vdisk set.

"hardwareType":"*Hardware details*"
The type of hardware used.

"memoryAvailable": "*size*"
The memory size available per server.

"memoryRequired": "*size*"
The memory size required per server.

"raidCode": "*Code*"
The RAID code of a vdisk set definition.

"storagePool": "*Storage pool* "
IBM Storage Scale file system storage pool for the vdisk NSDs. If the NSD usage is `dataAndMetadata` or `metadataOnly`, then the default storage pool must be `"system"` and need not be specified. If the NSD usage is `dataOnly`, the storage pool must be specified and cannot be `"system"`.

"totalRawSize": "*size*"
The total vdisk set size that is used from the declustered array, including the RAID code redundancy.

"usage": "*dataAndMetadata | metadataOnly | dataOnly*"
The NSD pool usage of the vdisk set. The default value is `dataAndMetadata`.

"vdiskCount": "*Count of Vdisk sets*"
The number of vdisks in the vdisk sets.

"vdiskRawSize": "*Usage details*"
The raw disk space that is used by each vdisk in the vdisk set.

"vdiskSetServerMemory":*Server memory size*
The server memory size available for the vdisk set.

"vdiskSize": "*size*"
The usable size of each vdisk in the vdisk set of the file system.

""nodeClass"": "*name*"
The recovery group server node class.

Examples

The following example gets the vdisk set information.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:text/html' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/diskmgmt/vdiskset'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  },
  "vdiskList": [
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs1",
      "memoryAvailable": 0,
      "recoveryGroups": "rgL",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "vd2_rgL"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs1",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "vd2_rgR"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs2",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "gpfs2_system_0"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs2",
      "memoryAvailable": 0,
      "recoveryGroups": "rgL",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
```

```
      "vdiskSize": 0,
      "vdisksetName": "gpfs2_system_1"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs0",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "gpfs0_system_2"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs0",
      "memoryAvailable": 0,
      "recoveryGroups": "rgL",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "gpfs0_system_1"
    },
    {
      "blocksize": 0,
      "created": "no",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "RG002VS001"
    },
    {
      "blocksize": 0,
      "created": "no",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "n005p002"
    },
    {
      "blocksize": 0,
      "created": "no",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "n005p003"
    },
    {
      "blocksize": 0,
      "created": "no",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "",
      "memoryAvailable": 0,
      "recoveryGroups": "rgR",
      "totalRawSize": 0,
```

```
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "n005p004"
    },
    {
      "blocksize": 0,
      "created": "yes",
      "freePercent": 0,
      "freeRawSize": 0,
      "fsName": "gpfs3",
      "memoryAvailable": 0,
      "recoveryGroups": "rgL",
      "totalRawSize": 0,
      "vdiskCount": 0,
      "vdiskRawSize": 0,
      "vdiskSize": 0,
      "vdisksetName": "vd4"
    },
  ]
}
```

Gnr/recoverygroups: GET

Gets a list of recovery groups that are configured in the cluster.

Availability

Available with the Elastic Storage Server.

Description

The GET `gnr/recoverygroups` request gets a list of recovery groups that are configured in the cluster. For more information about the fields in the data structures that are returned, see "mmlsrecoverygroup command" on page 327.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups
```

where

gnr/recoverygroups
Specifies that recovery groups are the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 33. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	

Table 33. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "recoveryGroups": [
      {
            "name":"Name"
            "declusteredArrayCount":"Number",
            "capacity":"Capacity",
            "freeSpace":"Available space",
            "usedSpace":"Used space",
            "nodes":"List of nodes"
            "oid":"ID",
            "vdiskCount":"Number of Vdisks",
            "pdiskCount":"Number of Pdisks",
      }
   ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"recoveryGroups":
Details of the recovery group.

"name":"*Name*"
The name of the recovery group.

"declusteredArrayCount":"*Number*"
Number of declustered arrays present in the recovery group.

"capacity":"*Capacity*"
Capacity of the recovery group.

"freeSpace":"*Available space*"
Available space that is left in the recovery group.

"usedSpace":"*Used space*"
The space used in the recovery group.

"nodes":"*List of nodes*"
The nodes that are part of the recovery group.

"oid":"*ID*"
The internal identifier of the recovery group that is used for paging.

"vdiskCount":"*Number of Vdisks*"
Number of Vdisks that are part of the recovery group.

"pdiskCount":"*Number of Pdisks*"
Number of Pdisks that are part of the recovery group.

Examples

The following example gets information about the recovery groups that are configured in the cluster.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/gnr/recoverygroups"
  },
  "recoveryGroups": [
    {
      "name": "RG1",
      "declusteredArrayCount": 2,
      "capacity": 12878459437056,
      "freeSpace": 0,
      "usedSpace": 0,
      "nodes": ["node1"],
      "oid": 0,
      "vdiskCount": 0,
      "pdiskCount": 24
    }
  ]
}
```

"mmlsrecoverygroup command" on page 327
Lists information about IBM Storage Scale RAID recovery groups.

"mmcrrecoverygroup command" on page 288
Creates an IBM Storage Scale RAID recovery group and its component declustered arrays and pdisks and specifies the servers.

Gnr/recoverygroups/{recoveryGroupName}: GET

Gets the details of a specific recovery group.

Availability

Available with the Elastic Storage Server.

Description

The GET `gnr/recoverygroup/{recoveryGroupName}` request gets the details of a specific recovery group that is configured in the cluster. For more information about the fields in the data structures that are returned, see "mmlsrecoverygroup command" on page 327.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups/recoveryGroupName
```

where

gnr/recoverygroups/recoveryGroupName
Specifies a particular recovery group as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 34. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
recoveryGroupName	The name of the recovery group about which you need the details.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "recoveryGroups": [
      {
            "name":"Name"
            "declusteredArrayCount":"Declustered array count",
            "capacity":"Capacity",
            "freeSpace":"Available space",
            "usedSpace":"Used space",
            "nodes":"List of nodes"
            "oid":"ID",
```

```
            "vdiskCount":"Number of Vdisks",
            "pdiskCount":"Number of Pdisks",
        }
    ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"recoveryGroups":
Details of the recovery group.

"name":"*Name*"
The name of the recovery group.

"declusteredArrayCount":"*Number*"
Number of declustered arrays present in the recovery group.

"capacity":"*Capacity*"
Capacity of the recovery group.

"freeSpace":"*Available space*"
Available space that is left in the recovery group.

"usedSpace":"*Used space*"
The space used in the recovery group.

"nodes":"*List of nodes*"
The nodes that are part of the recovery group.

"oid":"*ID*"
The internal identifier of the recovery group that is used for paging.

"vdiskCount":"*Number of Vdisks*"
Number of Vdisks that are part of the recovery group.

"pdiskCount":"*Number of Pdisks*"
Number of Pdisks that are part of the recovery group.

Examples

The following example gets information about the recovery group *RG1* that is configured in the cluster.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/RG1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
```

```
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/gnr/recoverygroups/RG1"
  },
  "recoveryGroups": [
    {
      "name": "RG1",
      "declusteredArrayCount": 2,
      "capacity": 12878459437056,
      "freeSpace": 0,
      "usedSpace": 0,
      "nodes": ["node1", "node2"],
      "oid": 0,
      "vdiskCount": 0,
      "pdiskCount": 24
    }
  ]
}
```

"mmlsrecoverygroup command" on page 327
Lists information about IBM Storage Scale RAID recovery groups.

"mmcrrecoverygroup command" on page 288
Creates an IBM Storage Scale RAID recovery group and its component declustered arrays and pdisks and specifies the servers.

Gnr/recoverygroups/{recoveryGroupName}/pdisks: GET

Gets the details of the physical disks that are available in a particular recovery group.

Availability

Available with the Elastic Storage Server.

Description

The GET `gnr/recoverygroup/{recoveryGroupName}/pdisks` request gets the details of the pdisks that are available in a recovery group. For more information about the fields in the data structures that are returned, see "mmlspdisk command" on page 324.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups/recoveryGroupName/pdisks
```

where

gnr/recoverygroups/recoveryGroupName
Specifies the recovery group to which the pdisks belong. Required.

pdisks
Specifies pdisks as the source of the GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 35. List of request parameters		
Parameter name	**Description and applicable keywords**	**Required/optional**
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
recoveryGroupName	The name of the recovery group to which the pdisks belong.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "pdisks": [
      {
          {
            "details":"Pdisk details"
             {
            "replacementPriority":"Replacement priority"
            "fru":"Field replaceable unit",
            "location":"Location",
            "wwn":"World wide name",
            "server":"Server",
            "rgIndex":"Recovery group index"
            "vendor":"Vendor name",
            "product":"Product name",
            "revision":"Revision",
            "serial":"Serial",
            "hardwareType":"Hardware type",
            "speed":"Speed",
      }
          "metrics":"Performance metrics"
             {
            "reads":"Reads"
            "writes":"Writes",
            "bytesRead":"Bytes read",
            "bytesWritten":"Bytes written",
            "ioErrors":"Number of I/O errors",
            "ioTimeOuts":"Number of I/O timeouts"
            "mediaErrors":"Media errors",
            "checksumErrors":"Cehcksum errors",
            "pathErrors":"Path errors",
            "relativePerformance":"Relative performance"
            "dataBadness":"Data badness",
      }
      "paths":"Path details"
             {
            "paths":"List of paths"
            "noOfPathsActive":"Number of active paths",
            "noOfPathsTotal":"Total number of paths",
            "expectedPathsActive":"Expected acive paths",
            "expectedPathsTotal":"Expected total paths",
            "noOfPathsActiveWrong":"Number of active wrong paths"
            "noOfPathsTotalWrong":"Total number of wrong paths",
```

```
            }
             "oid":"ID"
             "name":"Name",
             "declusteredArray":"Declustered array",
             "recoveryGroup":"Recovery group",
             "state":"State",
             "stateInfo":"State information"
             "healthState":"Health state",
             "freeSpace":"Free space",
             "capacity":"Capacity"
             "pathCount":"Path count",
   }
  ]
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"pdisks":"*Physical disk details*"
An array of information that lists the details of the pdisk.

"details":"*Pdisk details*"
Details of pdisk.

"replacementPriority":"*Replacement priority*"
Replacement priority that is defined for the pdisk.

"fru":"*Field replaceable unit*",
The field replaceable unit (FRU) of the pdisk.

"location":"*Location*",
Location of the pdisk in an enclosure.

"wwn":"*World wide name*",
The world wide name of the pdisk.

"server":"*Server*",
The server name of the ESS.

"rgIndex":"*Recovery group index*"
Recovery group index of the pdisk.

"vendor":"*Vendor name*",
Vendor name of the pdisk.

"product":"*Product name*",
Product name of the pdisk.

"revision":"*Revision*",
Revision of the pdisk.

"serial":"*Serial*",
The serial of the pdisk.

"hardwareType":"*Hardware type*",
Hardware type of the pdisk.

"speed":"*Speed*"
Speed of the pdisk.

"metrics":"*Performance metrics*"

"reads":"*Reads*"
Number of reads from the pdisk.

"writes":"*Writes*",
Number of writes to the pdisk.

"bytesRead":"*Bytes read*",
Number of bytes read from the pdisk.

"bytesWritten":"*Bytes written*",
Number of writes to the pdisk.

"ioErrors":"*Number of I/O errors*",
Number of I/O errors reported for the pdisk.

"ioTimeOuts":"*Number of I/O timeouts*"
Number of I/O timeouts reported for the pdisk.

"mediaErrors":"*Media errors*",
Number of media errors reported for the pdisk.

"checksumErrors":"*Cehcksum errors*",
Number of checksum errors reported for the pdisk.

"pathErrors":"*Path errors*",
Number of path errors reported for the pdisk.

"relativePerformance":"*Relative performance*"
Relative performance of the pdisk.

"dataBadness":"*Data badness*"
Data badness of the pdisk.

"paths":"*Path details*"

"paths":"*List of paths*"
List of paths of the pdisk.

"noOfPathsActive":"*Number of active paths*",
The number of active paths for the pdisk

"noOfPathsTotal":"*Total number of paths*",
The total number of paths for the pdisk.

"expectedPathsActive":"*Expected acive paths*",
The number of expected active paths for the pdisk.

"expectedPathsTotal":"*Expected total paths*",
The total number of expected paths for the pdisk.

"noOfPathsActiveWrong":"*Number of active wrong paths*"
The number of active wrong paths for the pdisk.

"noOfPathsTotalWrong":"*Total number of wrong paths*"
The total number of wrong paths for the pdisk.

"oid":"*ID*"
The ID used for paging.

"name":"*Name*",
Name of the pdisk.

"declusteredArray":"*Declustered array*",
The name of the declustered array of the pdisk.

"recoveryGroup":"*Recovery group*",
The name of the recovery group of the pdisk.

"state":"*State*",
The state of the pdisk

"stateInfo":"*State information*"
The state information about the pdisk.

"healthState":"*Health state*",
The health state of the pdisk.

"freeSpace":"*Free space*",
The free space of the pdisk in bytes.

"capacity":"*Capacity*"
The capacity of the pdisk in bytes.

"pathCount":"*Path count*"
The number of paths for the pdisk.

Examples

The following example gets information about the pdisks that are available in the recovery group *RG1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/RG1/pdisks
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "pdisks": [
    {
      "details": {
        "replacementPriority": 1000,
        "fru": "QEMU HARDDISK",
        "location": "24",
        "wwn": "naa.5000CCA01C514D48",
        "server": "gss-22.localnet.com",
        "rgIndex": 23,
        "vendor": "QEMU",
        "product": "QEMU HARDDISK",
        "revision": "1.5",
        "serial": "48",
        "hardwareType": "ROTATING",
        "speed": 7200
      },
      "metrics": {
        "reads": 3484,
        "writes": 698,
        "bytesRead": 546534588,
        "bytesWritten": 612032839,
        "ioErrors": 0,
        "ioTimeOuts": 0,
        "mediaErrors": 0,
        "checksumErrors": 0,
        "pathErrors": 0,
        "relativePerformance": 0.968,
        "dataBadness": 0
      },
      "paths": {
        "paths": "//gss-22/dev/sdar",
        "noOfPathsActive": 1,
        "noOfPathsTotal": 2,
        "expectedPathsActive": 1,
        "expectedPathsTotal": 2,
        "noOfPathsActiveWrong": 0,
```

```
        "noOfPathsTotalWrong": 0
      },
      "oid": 0,
      "name": "pdisk1",
      "declusteredArray": "DA1",
      "recoveryGroup": "RG1",
      "state": "NORMAL",
      "stateInfo": "ok",
      "healthState": "HEALTHY",
      "freeSpace": 533649686528,
      "capacity": 536602476544,
      "pathCount": 2
    }
  ]
}
```


"mmlspdisk command" on page 324
Lists information for one or more IBM Storage Scale RAID pdisks.

Gnr/recoverygroups/{recoveryGroupName}/pdisks/{pdiskName}: GET

Gets the details of a specific physical disk that is available in a recovery group.

Availability

Available with the Elastic Storage Server.

Description

The `GET gnr/recoverygroup/{recoveryGroupName}/pdisks/{pdiskName}` request gets the details of a specific pdisk that is part of a particular recovery group. For more information about the fields in the data structures that are returned, see "mmlspdisk command" on page 324.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups/recoveryGroupName/
pdisks/pdiskName
```

where

gnr/recoverygroups/recoveryGroupName
Specifies a particular recovery group that contains the pdisk. Required.

/pdisks/pdiskName
Specifies a particular pdisk as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 36. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Table 36. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
recoveryGroupName	The name of the recovery group that contains the pdisk.	Required.
pdiskName	The name of the pdisk about which you need the details.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "pdisks": [
      {
          {
            "details":"Pdisk details"
             {
            "replacementPriority":"Replacement priority"
            "fru":"Field replaceable unit",
            "location":"Location",
            "wwn":"World wide name",
            "server":"Server",
            "rgIndex":"Recovery group index"
            "vendor":"Vendor name",
            "product":"Product name",
            "revision":"Revision",
            "serial":"Serial",
            "hardwareType":"Hardware type",
            "speed":"Speed",
      }
          "metrics":"Performance metrics"
            {
            "reads":"Reads"
            "writes":"Writes",
            "bytesRead":"Bytes read",
            "bytesWritten":"Bytes written",
            "ioErrors":"Number of I/O errors",
            "ioTimeOuts":"Number of I/O timeouts"
            "mediaErrors":"Media errors",
            "checksumErrors":"Cehcksum errors",
            "pathErrors":"Path errors",
            "relativePerformance":"Relative performance"
            "dataBadness":"Data badness",
      }
      "paths":"Path details"
             {
            "paths":"List of paths"
            "noOfPathsActive":"Number of active paths",
            "noOfPathsTotal":"Total number of paths",
            "expectedPathsActive":"Expected acive paths",
            "expectedPathsTotal":"Expected total paths",
            "noOfPathsActiveWrong":"Number of active wrong paths"
             "noOfPathsTotalWrong":"Total number of wrong paths",
           }
```

```
            "oid":"ID"
            "name":"Name",
            "declusteredArray":"Declustered array",
            "recoveryGroup":"Recovery group",
            "state":"State",
            "stateInfo":"State information"
            "healthState":"Health state",
            "freeSpace":"Free space",
            "capacity":"Capacity"
            "pathCount":"Path count",
  }
 ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"pdisks":"*Physical disk details*"
Array of information that lists the details of the pdisk.

"details":"*Pdisk details*"
Details of pdisk.

"replacementPriority":"*Replacement priority*"
Replacement priority defined for the pdisk.

"fru":"*Field replaceable unit*",
The field replaceable unit (FRU) of the pdisk.

"location":"*Location*",
Location of the pdisk in an enclosure.

"wwn":"*World wide name*",
The world wide name of the pdisk.

"server":"*Server*",
The server name of the ESS.

"rgIndex":"*Recovery group index*"
Recovery group index of the pdisk.

"vendor":"*Vendor name*",
Vendor name of the pdisk.

"product":"*Product name*",
Product name of the pdisk.

"revision":"*Revision*",
Revision of the pdisk.

"serial":"*Serial*",
The serial of the pdisk.

"hardwareType":"*Hardware type*",
Hardware type of the pdisk.

"speed":"*Speed*"
Speed of the pdisk.

"metrics":"*Performance metrics*"

"reads":"*Reads*"
Number of reads from the pdisk.

"writes":"*Writes*",
Number of writes to the pdisk.

"bytesRead":"*Bytes read*",
Number of bytes read from the pdisk.

"bytesWritten":"*Bytes written*",
Number of writes to the pdisk.

"ioErrors":"*Number of I/O errors*",
Number of I/O errors reported for the pdisk.

"ioTimeOuts":"*Number of I/O timeouts*"
Number of I/O timeouts reported for the pdisk.

"mediaErrors":"*Media errors*",
Number of media errors reported for the pdisk.

"checksumErrors":"*Cehcksum errors*",
Number of checksum errors reported for the pdisk.

"pathErrors":"*Path errors*",
Number of path errors reported for the pdisk.

"relativePerformance":"*Relative performance*"
Relative performance of the pdisk.

"dataBadness":"*Data badness*"
Data badness of the pdisk.

"paths":"*Path details*"

"paths":"*List of paths*"
List of paths of the pdisk.

"noOfPathsActive":"*Number of active paths*",
The number of active paths for the pdisk

"noOfPathsTotal":"*Total number of paths*",
The total number of paths for the pdisk.

"expectedPathsActive":"*Expected acive paths*",
The number of expected active paths for the pdisk.

"expectedPathsTotal":"*Expected total paths*",
The total number of expected paths for the pdisk.

"noOfPathsActiveWrong":"*Number of active wrong paths*"
The number of active wrong paths for the pdisk.

"noOfPathsTotalWrong":"*Total number of wrong paths*"
The total number of wrong paths for the pdisk.

"oid":"*ID*"
The ID used for paging.

"name":"*Name*",
Name of the pdisk.

"declusteredArray":"*Declustered array*",
The name of the declustered array of the pdisk.

"recoveryGroup":"*Recovery group*",
The name of the recovery group of the pdisk.

"state":"*State*",
The state of the pdisk

"stateInfo":"*State information*"
The state information about the pdisk.

"healthState":"*Health state*",
The health state of the pdisk.

"freeSpace":"*Free space*",
The free space of the pdisk in bytes.

"capacity":"*Capacity*"
The capacity of the pdisk in bytes.

"pathCount":"*Path count*"
The number of paths for the pdisk.

Examples

The following example gets information about the pdisk *pdisk1* that is available in the recovery group *RG1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/RG1/pdisks/pdisk1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "pdisks": [
    {
      "details": {
        "replacementPriority": 1000,
        "fru": "QEMU HARDDISK",
        "location": "24",
        "wwn": "naa.5000CCA01C514D48",
        "server": "gss-22.localnet.com",
        "rgIndex": 23,
        "vendor": "QEMU",
        "product": "QEMU HARDDISK",
        "revision": "1.5",
        "serial": "48",
        "hardwareType": "ROTATING",
        "speed": 7200
      },
      "metrics": {
        "reads": 3484,
        "writes": 698,
        "bytesRead": 546534588,
        "bytesWritten": 612032839,
        "ioErrors": 0,
        "ioTimeOuts": 0,
        "mediaErrors": 0,
        "checksumErrors": 0,
        "pathErrors": 0,
        "relativePerformance": 0.968,
        "dataBadness": 0
      },
      "paths": {
        "paths": "//gss-22/dev/sdar",
        "noOfPathsActive": 1,
        "noOfPathsTotal": 2,
        "expectedPathsActive": 1,
        "expectedPathsTotal": 2,
```

```
        "noOfPathsActiveWrong": 0,
        "noOfPathsTotalWrong": 0
      },
      "oid": 0,
      "name": "pdisk1",
      "usteredArray": "DA1",
      "recoveryGroup": "RG1",
      "state": "NORMAL",
      "stateInfo": "ok",
      "healthState": "HEALTHY",
      "freeSpace": 533649686528,
      "capacity": 536602476544,
      "pathCount": 2
    }
  ]
}
```

“mmlspdisk command” on page 324
Lists information for one or more IBM Storage Scale RAID pdisks.

Gnr/recoverygroups/pdisk/change : PUT

Changes the runtime state of a physical disk (pdisk). The disk is resumed or suspended depending on its current state.

Availability

Available on all IBM Storage Scale Erasure Code editions.

Description

The PUT `scalemgmt/v2/gnr/recoverygroups/pdisk/change` request changes the runtime state of a physical disk (disk). The state is resumed or suspended based on its status. For more information about the fields in the data structures that are returned, see the “mmvdisk command” on page 341

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/gnr/recoverygroups/pdisk/
change
```

where

pdisk/change
Specifies `pdisk/change` as the resource. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes is available in the request data:

```
{
  "pdiskChangeArg": {
    "recoveryGroup": "string",
    "pdisk": "string",
    "identity": true | false,
    "clearErrorCounters": true | false,
    "diagnose": true | false,
    "suspend": true | false,
    "resume": true | false,
    "revive": true | false,
    "reviveFailing": true | false,
    "reviveSlow": true | false,
    "beginServiceDrain": true | false,
    "endServiceDrain": true | false,
```

```
    "simulateDead": true | false,
    "simulateFailing": true | false  }
}
```

The details of the parameters are given in the following list:

"pdiskChangeArg":
The parameters that are required to initiate the pdisk change process.

"recoveryGroup": "*recovery group name*"
Specifies the recovery group for the target pdisk.

"pdisk": "*pdisk name*"
Specifies the name of the physical disk.

"identity": true | false
Specifies whether the identity of the pdisk is authenticated.

"clearErrorCounters": true | false
Specifies whether the error counters for the pdisk are resolved.

"diagnose": true | false
Specifies whether basic tests can be run on the pdisk.

"suspend": true | false
Specifies whether the pdisk is suspended. If a pdisk remains in the suspended state longer than the predefined timeout period, IBM Storage Scale RAID begins rebuilding the data from that pdisk into spare space.

"resume": true | false
Specifies whether the previously suspended pdisk use is resumed.

"revive": true | false
Specifies whether the pdisk is made usable again by removing the pdisk state flags, *dead*, *failing*, and *readonly*.

"reviveFailing": true | false
Specifies that the pdisks that are marked with the *failing* state might be reused with the help of added diagnostic inputs.

"reviveSlow": true | false
Specifies that the pdisks that are marked with the *slow* state might be reused with the help of added diagnostic inputs.

"beginServiceDrain": true | false
Specifies whether all data can be drained out of the pdisk before it is temporarily removed.

"endServiceDrain": true | false
Specifies whether the drained data can be reloaded on to the pdisk.

"simulateDead": true | false
Specifies whether the `simulatedDead` state can be enabled to forcefully fail the pdisk.

"simulateFailing": true | false
Specifies whether the `simulatedFailing` state can be enabled to forcefully fail the pdisk.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": "Time when Job ran",
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
```

```
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the time at which the job was submitted.

"completed": "*Time*"
Specifies the time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
An array of commands that are run in this job.

"pids": *list*
Specifies a list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example changes the runtime status of the recovery group *rgL*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d
'{"pdiskChangeArg":{ \
 "recoveryGroup": "rgL",\
 "pdisk": "e1d1s02",\
 "identity": false,\
 "clearErrorCounters": false,\
 "diagnose": false, \
 "suspend": false, \
 "resume": true,\
 "revive": false, \
 "reviveFailing": false, \
 "reviveSlow": false,\
 "beginServiceDrain": false, \
 "endServiceDrain": false,\
 "simulateDead": false, \
 "simulateFailing": false \
```

```
} \
}' 'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/pdisk/change'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 4000000000004,
      "status": "RUNNING",
      "submitted": "2021-04-27 06:31:37,838",
      "completed": "N/A",
      "runtime": 3,
      "request": {
        "type": "PUT",
        "url": "scalemgmt/v2/pdisk/change"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

"mmvdisk command" on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/recoverygroups/pdisk/replace : PUT

Replaces a failed physical disk (pdisk).

Availability

Available on all IBM Storage Scale Erasure Code editions.

Description

The PUT `gnr/recoverygroups/pdisk/replace` request replaces a failed physical disk (pdisk). For more information about the fields in the data structures that are returned, see "mmvdisk command" on page 341.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/gnr/recoverygroups/pdisk/
replace
```

where

pdisk/replace
: Specifies `pdisk/replace` as the resource. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes are available in the request data:

```
{
  "pdiskReplaceArg": {
    "recoveryGroup": "Name of the recovery group",
    "prepare": true | false,
    "cancel": true | false,
    "force": true | false,
    "pdisk": "The pdisk name"
  }
}
```

The details of the parameters are given in the following list:

"pdiskReplaceArg":
The parameters that are required to initiate the pdisk replacement process.

"recoveryGroup": "*The recovery group name*"
Specifies the recovery group for the target pdisk.

"prepare": true | false
Specifies whether the disk is prepared for removal and replacement

"cancel": true | false
Specifies whether the preparation for the removal and replacement of the disk is canceled.

"force": true | false
Specifies whether the replacement disk is allowed to have a field replacement unit (FRU) or type that is different from the removed disk.

"pdisk": "*pdisk name*"
Specifies the name of the physical disk.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example replaces the failed disk in the recovery group named *rgR*.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
   "pdiskReplaceArg": { \
     "recoveryGroup": "rgR", \
     "prepare": false, \
     "cancel": false, \
     "force": false, \
     "pdisk": "e1d1s08" \
   } \
 }' 'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/pdisk/replace'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 4000000000001,
      "status": "RUNNING",
      "submitted": "2021-04-27 06:31:37,838",
      "completed": "N/A",
      "runtime": 3,
      "request": {
        "type": "PUT",
        "url": "scalemgmt/v2/gnr/recoverygroups/pdisk/replace"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
```

```
  }
}
```

Related reference

"mmvdisk command" on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/recoverygroups/{recoveryGroupName}/vdisks: GET

Returns the details of the virtual disks that are available in a specified recovery group.

Availability

Available with the Elastic Storage Server.

Description

The `GET gnr/recoverygroup/{recoveryGroupName}/vdisks` request gets the details of the vdisks that are part of a particular recovery group. For more information about the fields in the data structures that are returned, see "mmlsvdisk command" on page 337.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups/recoveryGroupName/vdisks
```

where

gnr/recoverygroups/recoveryGroupName
Specifies a particular recovery group that contains the vdisk. Required.

vdisks
Specifies the vdisks as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 37. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
recoveryGroupName	The name of the recovery group that contains the vdisks.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "vdisks": [
      {

            "name":"Name",
            "recoveryGroup":"Recovery group",
            "declusteredArray":"Declustered array",
            "size":"Size",
            "raidCode":"RAID code"
            "trackSize":"Track size",
            "checksumGranularity":"Checksum granularity",
            "remarks":"Remarks",
            "state":"State",
            "healthState":"Health status",
            "oid":"ID",
  }
 ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": "*ReturnCode*"
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"vdisks":"*Virtual disk details*"
Array of information that lists the details of the vdisk that is available in the specified recovery group.

"name":"*Name*"
Name of the vdisk.

"recoveryGroup":"*Recovery group*"
The name of the recovery group of the vdisk.

"declusteredArray":"*Declustered array*"
The name of the declustered array of the vdisk.

"size":"*Size*"
Size of the vdisk.

"raidCode":"*RAID code*"
RAID code of the vdisk.

"trackSize":"*Track size*"
Size of the track.

"checksumGranularity":"*Checksum granularity*"
The granularity of checksum.

"remarks":"*Remarks*"
Remarks about the vdisk.

"state":"*State*"
The state of the vdisk.

"healthState":"*Health status*"
The health status of the vdisk.

"oid":"*ID*"
The ID used for paging.

Examples

The following example gets information about the vdisks that are available in the recovery group *RG1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/RG1/vdisks
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "vdisks": [
    {
      "name": "vdisk1",
      "recoveryGroup": "RG1",
      "declusteredArray": "DA1",
      "size": 262144,
      "raidCode": "2WayReplication",
      "trackSize": 262144,
      "checksumGranularity": 32768,
      "remarks": "logTip",
      "state": "ok",
      "healthState": "UNKNOWN",
      "oid": 0
    }
  ]
}
```

“mmlsvdisk command” on page 337
Lists information for one or more IBM Storage Scale RAID vdisks.

“mmcrvdisk command” on page 291
Creates a vdisk within a declustered array of an IBM Storage Scale RAID recovery group.

Gnr/recoverygroups/{recoveryGroupName}/vdisks/{vdiskName}: GET

Returns the details a virtual disk that is available in a specified recovery group.

Availability

Available with the Elastic Storage Server.

Description

The GET gnr/recoverygroup/{recoveryGroupName}/vdisks/{vdiskName} request gets the details of a specific vdisk that is part of a particular recovery group. For more information about the fields in the data structures that are returned, see “mmlsvdisk command” on page 337.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/recoverygroups/recoveryGroupName/
vdisks/vdiskName
```

where

gnr/recoverygroups/recoveryGroupName
Specifies a particular recovery group that contains the vdisk. Required.

/vdisks/vdiskName
Specifies a particular vdisk as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 38. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma-separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
recoveryGroupName	The name of the recovery group that contains the vdisk.	Required.
vdiskName	The name of the vdisk about which you need the details.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   "vdisks": [
```

```
      {

            "name":"Name",
            "recoveryGroup":"Recovery group",
            "declusteredArray":"Declustered array",
            "size":"Size",
            "raidCode":"RAID code"
            "trackSize":"Track size",
            "checksumGranularity":"Checksum granularity",
            "remarks":"Remarks",
            "state":"State",
            "healthState":"Health status",
            "oid":"ID",
  }
 ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"vdisks":"*Virtual disk details*"
Array of information that lists the details of the vdisk that is available in the specified recovery group.

"name":"*Name*",
Name of the vdisk.

"recoveryGroup":"*Recovery group*"
The name of the recovery group of the vdisk.

"declusteredArray":"*Declustered array*"
The name of the declustered array of the vdisk.

"size":"*Size*"
Size of the vdisk.

"raidCode":"*RAID code*"
RAID code of the vdisk.

"trackSize":"*Track size*"
Size of the track.

"checksumGranularity":"*Checksum granularity*"
The granularity of checksum.

"remarks":"*Remarks*"
Remarks about the vdisk.

"state":"*State*"
The state of the vdisk.

"healthState":"*Health status*"
The health status of the vdisk.

"oid":"*ID*"
The ID used for paging.

Examples

The following example gets information about the vdisk *vdisk1* that is available in the recovery group *RG1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/recoverygroups/RG1/vdisks/vdisk1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "vdisks": [
    {
      "name": "vdisk1",
      "recoveryGroup": "RG1",
      "declusteredArray": "DA1",
      "size": 262144,
      "raidCode": "2WayReplication",
      "trackSize": 262144,
      "checksumGranularity": 32768,
      "remarks": "logTip",
      "state": "ok",
      "healthState": "UNKNOWN",
      "oid": 0
    }
  ]
}
```

“mmlsvdisk command” on page 337
Lists information for one or more IBM Storage Scale RAID vdisks.

“mmcrvdisk command” on page 291
Creates a vdisk within a declustered array of an IBM Storage Scale RAID recovery group.

Gnr/clustermgmt/nodes/{names}/state: GET

Gets the status of the GPFS daemon on a specified node of the cluster.

Availability

Available on all IBM Storage Scale editions.

Description

The `GET gnr/clustermgmt/nodes/{names}/state` request gets the details of the GPFS daemon on the node that is specified in the request parameter. For more information about the fields in the data structures that are returned, see the **mmgetstate** command in *IBM Storage Scale: Command and Programming Reference* in the IBM Storage Scale documentation.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/clustermgmt/nodes/{names}/state
```

where

nodes
Specifies the resource of this GET call. Required.

names
Specifies the node names. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
    "code": Status code,
    "message": "Status message"
  },
  "nodeState": [
    {
      "nodeName": "Node name",
      "number": Node number,
      "state": "UNKNOWN | ACTIVE | ARBITRATING | DOWN ",
      "quorum": Node quorum,
      "quorumUp": Number of quorum nodes up,
      "totalNode": Total number of nodes,
      "remarks": "Comments",
      "daemonNodeName": "Name of node"
      "daemonShortName": "Shortname of node"
      "cnfsState": "The GPFS daemon state"
    }
  ]
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"nodeState"
The state of the Nodes.

"nodeName":"*Node name*"
The name of the node.

"number": *Node number*
Specifies the number of the node.

"state":"*UNKNOWN | ACTIVE | ARBITRATING | DOWN* "
The status of GPFS daemon.

UNKNOWN
The state cannot be detected through the API request.

ACTIVE
The GPFS daemon is ready for operation.

ARBITRATING
A node is trying to form a quorum with the other available nodes.

DOWN
The GPFS daemon is not running on the node or is recovering from an internal error.

"quorum":"*quorum nodes*"
The number of quorum nodes.

"quorumUp":"*quorum nodes that are up* "
The number of quorum nodes that are up.

"totalnode":"*All nodes*"
The total number of nodes.

"remarks":"*comments*"
The additional comments on the node status.

"daemonNodeName":"*Node name*"
The name of the daemon node.

"daemonShortName":"*Node shortname*"
The shortname of the daemon node.

"cnfsState":"*GPFS daemon state*"
The state of the GPFS daemon.

Examples

The following example gets information on the status of the GPFS daemon on the node `hciece01`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/clustermgmt/nodes/hciece01/state
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "nodeStates": [
    {
      "nodeName": "hciece01",
      "daemonNodeName": "rhel-41.openstacklocal",
      "daemonShortName": "rhel-41",
      "number": 0,
      "state": "UNKNOWN",
      "quorum": 3,
      "quorumUp": 2,
      "totalNode": 3,
      "remarks": " ",
      "cnfsState": " "
    }
  ]
}
```

Related information:

mmgetstate command in the IBM Storage Scale documentation.

Gnr/clustermgmt/server/configure: POST

Configures a new node or an mmvdisk node class for IBM Storage Scale RAID servers.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `Gnr/clustermgmt/server/configure` request creates and configures a new node or an mmvdisk node class for the IBM Storage Scale RAID servers. The nodes or node class must not already be configured or must not be serving a recovery group. For more information about the fields in the data structures that are returned, see the "mmvdisk server command" on page 350.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/clustermgmt/server/
configure
```

Request headers

```
Accept: application/json
```

Request data

The following list of attributes are available in the request data:

```
{
  "node": "Node name",
  "recycle": "none | one| all | Number",
  "pagePool": "nM | nG | n%",
  "maxBlockSize": "2M |4M |8M |16M",
}
```

The details of the parameters are given in the following list:

"node": ***The name of the node***
: The node name that is to be configured.

"recycle": ***none | one | all | Number***
: Specifies the number of nodes that can be simultaneously restarted so that IBM Storage Scale configuration changes can take effect.

  none
  : This is the default value and indicates that the administrator must use the commands **mmstartup** and **mmshutdown** to recycle the nodes or node class.

  one
  : Specifying this value indicates that recycling occurs one node at a time .

  all
  : Specifying this value indicates that all specified nodes are recycled simultaneously.

  Number
  : Specifying a number indicates that the defined number of nodes can be chosen to be recycled at the same time.

"pagepool": ***nM | nG | n%***
: The IBM Storage Scale page pool value. This option should only be used under instruction from IBM.

"maxBlockSize" : ***2M |4M |8M|16M***
: The maximum file system block size to be used with IBM Storage Scale RAID.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
```

```
          {
            "commands":"String",
            "progress":"String,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Date and Time",
        "completed":"Date and Time",
        "runtime":Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
  ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and the time at which the job was submitted.

"completed":"*Date and Time*"
Specifies the date and the time at which the job was completed.

"runtime":"*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to configure a new node named `ess-11` .

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/clustermgmt/server/configure'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 563,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/gnr/clustermgmt/server/configure"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk server configure {--node-class NcName | -N Node[,Node...]}
                          [--recycle {none |one |all | Number}]
                          [--pagepool {nM |nG | n%}]
                          [--maxblocksize {2M | 4M | 8M | 16M}]
                          [--update]" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk server command” on page 350
Manages mmvdisk recovery group servers for IBM Storage Scale RAID.

Gnr/diskmgmt/vdiskset/define: POST

Defines uniform virtual disk (vdisk) sets across all IBM Storage Scale RAID recovery groups.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `gnr/diskmgmt/vdiskset/define` request defines vdisk sets for all recovery groups that belong to IBM Storage Scale RAID. For more information about the fields in the data structures that are returned, see the "mmvdisk vdiskset command" on page 380.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/diskmgmt/vdiskset/define
```

where:

vdiskset/define
Specifies `vdiskset/define` as the resource. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes is available in the request data:

```
{
  "name": "string",
  "declusteredArray": "string",
  "pool": "string",
  "usableSize": size,
  "raidCode": "string",
  "nsdUsageType": "string",
  "blockSize": "string",
  "recoveryGroups": [
    "string"
  ],
   "existingVdiskSet": "Name",
   "forceFlag": true | false
}
```

The details of the parameters are given in the following list:

"name": *Vdisk set name*
The name of the vdisk set to be defined.

"declusteredArray": "*Declustered array name*"
The name of a single declustered array that must be present in each of the recovery groups for the vdisk set definition. The member vdisk NSDs is defined in this declustered array. You can skip specifying this name if the recovery group has only a single user declustered array. For example, "*DA1*"

"pool": "*storage pool*"
The name of the file system storage pool for the vdisk set.

"usableSize": Size
The size that is available for use for the vdisk set definition.

"raidCode": "*RAID code*"
The RAID code that is used by the member vdisk NSDs.

"nsdUsageType": "NSD usage"
The IBM Storage Scale file system data usage for the vdisk NSDs. The default value is `dataAndMetadata`. The other valid values are `metadataOnly` and `dataOnly`.

"blockSize": "*Block size usage*"
The block size that is used by the member vdisk NSDs. Valid values for 3WayReplication and 4WayReplication are `256K, 512K, 1M, and 2M`. The valid values for 4+2P and 4+3P are: `512K, 1M, 2M, 4M, 8M`. The valid values for 8+2P and 8+3P are: `512K, 1M, 2M, 4M, 8M, 16m`.

"recoveryGroups": "list"
The list of recovery groups where the vdisk is to be defined.

"existingVdiskSet": "Name"
The name of the existing vdisk set from which the attribute definition is copied.

"forceFlag": true | false
Specifies whether the `forceFlag` is enabled to copy the vdisk set definition into an incompatible declustered array.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands used",
             "progress":"Request progress,
             "exitCode":"Exit code",
             "stderr":"STD error messages",
             "stdout":"CLI messgaes",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         "pids":"Process IDs
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"stdout":"*CLI messages*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*""
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

"pids":"Process ID"
The process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to define the vdisk set `vs1` in the recovery group `RG1`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' -d {
  "name": "vs1",\
  "declusteredArray": "DA1",\
  "pool": "system",\
  "usableSize": 10,\
  "raidCode": "4+2p",\
  "nsdUsageType": "dataAndMetadata",\
  "blockSize": "8M",\
  "recoveryGroups": [ \
    "RG1" \
  ],\
  "existingVdiskSet": "vs2",\
  "forceFlag": false
}' 'https://198.51.100.1:443/scalemgmt/v2/gnr/diskmgmt/vdiskset/define'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/gnr/diskmgmt/vdiskset/define"
      "data": "{\"define\":{\"vdiskset\":\"vs1\",\"owner\":\"root\",\"path\":\"/mnt/gpfs0/
rest1001\",\"permissions\":\"555\"}"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk vdiskset define  --vdisk-set VdiskSet
                        --recovery-group {all | RgName[,RgName...]}
                        --code RaidCode  --block-size BlockSize
                        --set-size {n% | n | nK | nM | nG | nT}
                        [--declustered-array DaName]
                        [--nsd-usage NsdUsage [--storage-pool StoragePool]] " ],
```

```
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk vdiskset command” on page 380
Manages mmvdisk vdisk sets for IBM Storage Scale RAID.

Gnr/recoverygroups/{recoveryGroupName}/node/{name}: POST

Adds a node to an existing recovery group.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `gnr/recoverygroups/{recoveryGroupName}/node/{name}` request adds a node to an existing recovery group. For more information about the fields in the data structures that are returned, see the “mmvdisk recoverygroup command” on page 357.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/recoverygroups/
recoveryGroupName/node/name
```

Request headers

```
Accept: application/json
```

Request data

No request data.

Parameters

Table 39. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
recoveryGroupName	Specifies the name of the recovery group where the node is added.	Required
name	Specifies the name of the node that is added.	Required

Response data

```
{
   "status": {
```

```
    "code":ReturnCode",
    "message":"ReturnMessage"
 },
 "jobs": [
    {
       "result":"",
         {
           "commands":"String",
           "progress":"String,
           "exitCode":"Exit code",
           "stderr":"Error",
           "stdout":"String",
         },
       "request":" ",
         {
           "type":"{GET | POST | PUT | DELETE}",
           "url":"URL",
           "data":"",
         }
       "jobId":"ID",
       "submitted":"Date and Time",
       "completed":Date and Time",
       "runtime":Duration",
       "status":"Job status",
       "pids":"Process IDs
       }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
An array of commands that are run in this job.

"progress":"*String*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to configure a new node named `ess-11` for the recovery group `rg1`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/diskmgmt/recoverygroups/rg1/node/ess-11'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 563,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/gnr/diskmgmt/recoverygroups/rg1/node/ess-11"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk recoverygroup add  --recovery-group RgName
                              -N Node [--match  N]
                              [--fanout N] [-v {yes | no}]" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk recoverygroup command” on page 357
Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Gnr/diskmgmt/vdiskset/create/{vdiskSet}: POST

Creates a member vdisk NSD for a vdisk set.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST Gnr/diskmgmt/vdiskset/create/{vdiskSet} request creates a member vdisk NSD for a vdisk set. For more information about the fields in the data structures that are returned, see the “mmvdisk command” on page 341.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/diskmgmt/vdiskset/create/
vdiskSet
```

where:

vdiskSet
Specifies the name of the vdisk set where the member vdisk NSD is created. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Parameters

Table 40. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
vdiskSet	The name of the vdisk set where the member vdisk NSD is created.	Required

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"commands used",
             "progress":"Request progress status,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"CLI message",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         "pids":"Process IDs
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

""result"

"commands":"*String*"
An array of commands that are run in this job.

"progress":"*String*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of the command. Zero is success and nonzero denotes failure.

"stderr":"*String*"
Specifies the CLI messages from stderr.

"stdout":"*String*""
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":" "
Specifies the request data. Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Time*"
Specifies the data and time at which the job was submitted.

"completed":*Time*"
Specifies the data and time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to define the member vdisk NSD in the vdisk set `vs1`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/diskmgmt/vdiskset/create/vs1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000004,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/gnr/diskmgmt/vdiskset/create/vs1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ mmvdisk vdiskset create  --vdisk-set {all | VdiskSet[,VdiskSet...]} ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk command” on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/diskmgmt/vdiskset/undefine/{vdiskSet}: DELETE

Removes a vdisk set definition from IBM Storage Scale RAID recovery groups.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The `DELETE gnr/diskmgmt/vdiskset/undefine/`*{vdiskSet}* request removes the definition of vdisk sets from recovery groups that belong to IBM Storage Scale RAID. For more information about the fields in the data structures that are returned, see the “mmvdisk command” on page 341.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/gnr/diskmgmt/vdiskset/
undefine/{vdiskSet}
```

where

{vdiskSet}
Specifies the vdisk set that must be removed from the recovery groups. Required.

Request headers

```
Accept: application/json
```

Parameters

Table 41. List of parameters		
Parameter name	**Description and applicable keywords**	**Required/Optional**
vdiskSet	Name of the vdisk set whose definition is to be removed.	Required

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"Commands issued ",
             "progress":"Request progress status,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"CLI messages",
           },
         "request":" ",
           {
             "type":"{GET | DELETE | DELETE | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "completed":Duration
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the link at the end of the topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*commnds issued* "
An array of commands that are issued in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the Exit code of the command. Zero indicates success, and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from *stderr*.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data. Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the data and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the data and time at which the job was completed.

"runtime": "*duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

Examples

The following example shows how to remove the vdisk set definition *vs1*:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/diskmgmt/vdiskset/undefine/vs1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/gnr/diskmgmt/vdiskset/undefine/vs1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk vdiskset undefine  --vdisk-set VdiskSet [--confirm]" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
```

```
  }
}
```

Related reference

“mmvdisk command” on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/diskmgmt/vdiskset/delete/{vdiskSet}: DELETE

Deletes member vdisk NSDs from a vdisk set.

Availability

Available with IBM Storage Scale Erasure Code Edition only

Description

The `DELETE gnr/diskmgmt/vdiskset/delete/{vdiskSet}` request deletes a member vdisk NSD from a vdisk set. For more information about the fields in the data structures that are returned, see the “mmvdisk command” on page 341.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/gnr/diskmgmt/vdiskset/
delete/{vdiskSet}
```

where

vdiskset/delete/*{vdiskSet}*
: Specifies the vdisk set from which the member NSDs must be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

Table 42. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
vdiskSet	Name of the vdisk set from which member vdisk NSDs are to be removed.	Required

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
```

```
        "result":"",
          {
            "commands":"Commands used",
            "progress":"Request progress,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | DELETE | DELETE | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
         "runtime":duration
        "status":"Job status",
        }
       ],
  }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success, and any value other than zero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr*.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":
Specifies the request data. Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"runtime":*duration*
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to delete member vdisk NSDs from the vdisk set *vs1*:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/diskmgmt/vdiskset/delete/vs1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/gnr/diskmgmt/vdiskset/undefine/vs1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk vdiskset delete --vdisk-set {all | VdiskSet[,VdiskSet...]} |
                        [--recovery-group  RgName[,RgName...]] [-p]" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk command” on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/diskmgmt/vdiskset/{vdiskSet}/rename/{newVdiskSet}: POST

Renames an existing vdisk set.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `Gnr/diskmgmt/vdiskset/{vdiskSet}/rename/{newVdiskSet}` request renames an existing vdisk set. For more information about the fields in the data structures that are returned, see the “mmvdisk command” on page 341.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/diskmgmt/vdiskset/
vdiskSet/rename/newVdiskSet
```

where:

vdiskSet
Specifies the vdisk set to be renamed. Required.

newVdiskSet
Specifies the new vdisk set name. Required.

Request headers

```
Accept: application/json
```

Request data

No request data

Parameters

Table 43. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
vdiskSet	The name of the vdisk set that needs to be renamed.	Required
newVdiskSet	The new name of the vdisk set.	Required

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
             "progress":"Request progress,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"CLI messages",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId": "ID",
         "submitted":"Time",
         "completed": "Time",
         "completed": "Time"
         "status": "Job status",
         "pids": "Process IDs"
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
An array of commands that are run in this job.

"progress":"*String*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data"
Specifies the request data. Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the date and time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to rename the vdisk set as vs2.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/diskmgmt/vdiskset/vs1/rename/vs2'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/gnr/diskmgmt/vdiskset/vs1/rename/vs2"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmvdisk vdiskset rename --vdisk-set VdiskSet --new-name NewName " ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmvdisk command” on page 341
Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Gnr/clustermgmt/{newNodeName}/service/{serviceName}: POST

Starts a new node on which the GPFS service can run.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The POST `gnr/clustermgmt/{newNodeName}/service/{serviceName}` request creates a new node for a GPFS service. For more information about the fields in the data structures that are returned, see the **mmstartup** command in the IBM Storage Scale documentation.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/gnr/clustermgmt/newNodeName/
service/serviceName
```

where:

newNodeName
Specifies the new node to be started. Required.

serviceName
Specifies the GPFS service for which the node is started. Required.

Request headers

```
Accept: application/json
```

Request data

No request data

Parameters

Table 44. List of parameters		
Parameter name	**Description and applicable keywords**	**Required/Optional**
newNodeName	The name of the new node that is to be started .	Required
serviceName	The name of the GPFS service for which the node is started.	Required

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         "pids":"Process IDs
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
An array of commands that are run in this job.

"progress":"*String*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of the command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data. Optional.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and the time at which the job was submitted.

"completed":*Time*"
Specifies the date and the time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

"pids":"Process ID"
The process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to start a new node es-13 for the GPFS service NFS .

Request data:

```
curl -k -u admin:admin001 -X POST --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/clustermgmt/es-13/service/NFS
'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 563,
    "request" : {
      "type" : "POST",
      "url" : "gnr/clustermgmt/es-13/service/NFS"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmstartup [-a | -N {Node[,Node...] | NodeFile | NodeClass}] [-E
EnvVar=value ...] " ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
```

```
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related information:

mmstartup command in the IBM Storage Scale documentation.

Gnr/recoverygroups/suspend/{recoveryGroupName}/{node}: PUT

Suspends a single node on a scale-out recovery group.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The PUT `gnr/recoverygroups/suspend/{recoveryGroupName}/{node}` request suspends a single node in a scale-out recovery group for maintenance purposes. For more information about the fields in the data structures that are returned, see the “mmvdisk recoverygroup command” on page 357.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/gnr/recoverygroups/suspend/
{recoveryGroupName}/{node}
```

where

recoveryGroupName
Specifies recovery group to which the suspended node belongs. Required.

node
Specifies the node of the recovery group that is suspended. Required.

Request headers

```
Accept: application/json
```

Request data

No request data:

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time ",
      "completed": "Date and time ",
      "runtime": Duration,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
```

```
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the Status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
Specifies the return message.

"code": *ReturnCode*
Specifies the return code.

Examples

The following example suspends the node *hciece01-hs.gpfs.net* of the recovery group *rg1*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/
json' --header 'Authorization: Basic a3JpdGlrYTE6YWRtaW4wMDE=' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/recoverygroups/suspend/rg1/hciece01-hs.gpfs.net'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 4000000000004,
      "status": "RUNNING",
```

```
      "submitted": "2021-04-27 06:31:37,838",
      "completed": "N/A",
      "runtime": 3,
      "request": {
        "type": "PUT",
        "url": "scalemgmt/v2/gnr/recoverygroups/suspend/rg1/hciece01-hs.gpfs.net"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

“mmvdisk recoverygroup command” on page 357
Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Gnr/recoverygroups/resume/{recoveryGroupName}/{node}: PUT

Resumes a suspended node for a specific recovery group.

Availability

Available with IBM Storage Scale Erasure Code Edition only.

Description

The PUT `gnr/recoverygroups/resume/{recoveryGroupName}/{node}` request resumes a suspended node for a recovery group. The request helps to restart IBM Storage Scale on the resumed node and also make its disks available. For more information about the fields in the data structures that are returned, see the “mmvdisk recoverygroup command” on page 357.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/gnr/recoverygroups/resume/
{recoveryGroupName}/{node}
```

where

recoverygroupName
: Specifies the recovery group to which the node to be resumed is associated. Required.

node
: Specifies the node that must be resumed.

Request headers

```
Accept: application/json
```

Request data

No request data:

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
```

```
      "submitted": "Date and time ",
      "completed": "Date and time ",
      "runtime": Duration,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
Specifies the return message.

"code": *ReturnCode*
Specifies the return code.

Examples

The following example resumes the node *hciece01-hs.gpfs.net* for the recovery group *rg2*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/
json' --header 'Authorization: Basic a3JpdGlrYTE6YWRtaW4wMDE=' 'https://198.51.100.1:443/
scalemgmt/v2/gnr/recoverygroups/resume/rg2/hciece01-hs.gpfs.net'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000001,
      "status": "RUNNING",
      "submitted": "2021-04-27 06:31:37,838",
      "completed": "N/A",
      "runtime": 12,
      "request": {
        "type": "PUT",
        "url": "scalemgmt/v2/gnr/recoverygroups/resume/rg2/hciece01-hs.gpfs.net"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

“mmvdisk recoverygroup command” on page 357
Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Appendix A. Best-practice recommendations for IBM Storage Scale RAID

This topic includes some best-practice recommendations for using IBM Storage Scale RAID.

Planning a IBM Storage Scale RAID implementation requires consideration of the nature of the JBOD arrays being used, the required redundancy protection and usable disk capacity, the required spare capacity and maintenance strategy, and the ultimate GPFS file system configuration.

- Assign a primary and backup server to each recovery group.

  Each JBOD array should be connected to two servers to protect against server failure. Each server should also have two independent paths to each physical disk to protect against path failure and provide higher throughput to the individual disks.

  Define multiple recovery groups on a JBOD array, if the architecture suggests it, and use mutually reinforcing primary and backup servers to spread the processing evenly across the servers and the JBOD array.

  Recovery group server nodes can be designated GPFS quorum or manager nodes, but they should otherwise be dedicated to IBM Storage Scale RAID and not run application workload.
- Configure recovery group servers with a large vdisk track cache and a large page pool.

  The `nsdRAIDTracks` configuration parameter tells IBM Storage Scale RAID how many vdisk track descriptors, not including the actual track data, to cache in memory.

  In general, a large number of vdisk track descriptors should be cached. The `nsdRAIDTracks` value for the recovery group servers should be on the order of 100000, or even more if server memory exceeds 128 GiB. If the expected vdisk NSD access pattern is random across all defined vdisks and within individual vdisks, a larger value for `nsdRAIDTracks` might be warranted. If the expected access pattern is sequential, a smaller value can be sufficient.

  The amount of actual vdisk data (including user data, parity, and checksums) that can be cached depends on the size of the GPFS page pool on the recovery group servers and the percentage of page pool reserved for IBM Storage Scale RAID. The `nsdRAIDBufferPoolSizePct` parameter specifies what percentage of the page pool should be used for vdisk data. The default is 80%, but it can be set as high as 90% or as low as 10%. Because a recovery group server is also an NSD server and the vdisk buffer pool also acts as the NSD buffer pool, the configuration parameter `nsdBufSpace` should be reduced to its minimum value of 10%.

  As an example, to have a recovery group server cache 20000 vdisk track descriptors (`nsdRAIDTracks`), where the data size of each track is 4 MiB, using 80% (`nsdRAIDBufferPoolSizePct`) of the page pool, an approximate page pool size of 20000 * 4 MiB * (100/80) ≈ 100000 MiB ≈ 98 GiB would be required. It is not necessary to configure the page pool to cache all the data for every cached vdisk track descriptor, but this example calculation can provide some guidance in determining appropriate values for `nsdRAIDTracks` and `nsdRAIDBufferPoolSizePct`.
- Define each recovery group with at least one large declustered array.

  A large declustered array contains enough pdisks to store the required redundancy of IBM Storage Scale RAID vdisk configuration data. This is defined as at least nine pdisks plus the effective spare capacity. A minimum spare capacity equivalent to two pdisks is strongly recommended in each large declustered array. The code width of the vdisks must also be considered. The effective number of non-spare pdisks must be at least as great as the largest vdisk code width. A declustered array with two effective spares where 11 is the largest code width (8 + 3p Reed-Solomon vdisks) must contain at least 13 pdisks. A declustered array with two effective spares in which 10 is the largest code width (8 + 2p Reed-Solomon vdisks) must contain at least 12 pdisks.
- Define the log vdisks based on the type of configuration.

See "Typical configurations" under “Log vdisks” on page 62 and Chapter 10, “Setting up IBM Storage Scale RAID on the Elastic Storage Server,” on page 115 for log vdisk considerations.

- Determine the declustered array maintenance strategy.

  Disks will fail and need replacement, so a general strategy of deferred maintenance can be used. For example, failed pdisks in a declustered array are only replaced when the spare capacity of the declustered array is exhausted. This is implemented with the replacement threshold for the declustered array set equal to the effective spare capacity. This strategy is useful in installations with a large number of recovery groups where disk replacement might be scheduled on a weekly basis. Smaller installations can have IBM Storage Scale RAID require disk replacement as disks fail, which means the declustered array replacement threshold can be set to 1.

- Choose the vdisk RAID codes based on GPFS file system usage.

  The choice of vdisk RAID codes depends on the level of redundancy protection required versus the amount of actual space required for user data, and the ultimate intended use of the vdisk NSDs in a GPFS file system.

  Reed-Solomon vdisks are more space efficient. An 8 + 3p vdisk uses approximately 27% of actual disk space for redundancy protection and 73% for user data. An 8 + 2p vdisk uses 20% for redundancy and 80% for user data. Reed-Solomon vdisks perform best when writing whole tracks (the GPFS block size) at once. When partial tracks of a Reed-Solomon vdisk are written, parity recalculation must occur.

  Replicated vdisks are less space efficient. A vdisk with 3-way replication uses approximately 67% of actual disk space for redundancy protection and 33% for user data. A vdisk with 4-way replication uses 75% of actual disk space for redundancy and 25% for user data. The advantage of vdisks with *N*-way replication is that small or partial write operations can complete faster.

  For file system applications where write performance must be optimized, the preceding considerations make replicated vdisks most suitable for use as GPFS file system `metadataOnly` NSDs, and Reed-Solomon vdisks most suitable for use as GPFS file system `dataOnly` NSDs. The volume of GPFS file system metadata is usually small (1% - 3%) relative to file system data, so the impact of the space inefficiency of a replicated RAID code is minimized. The file system metadata is typically written in small chunks, which takes advantage of the faster small and partial write operations of the replicated RAID code. Applications are often tuned to write file system user data in whole multiples of the file system block size, which works to the strengths of the Reed-Solomon RAID codes both in terms of space efficiency and speed.

  When segregating vdisk NSDs for file system `metadataOnly` and `dataOnly` disk usage, the `metadataOnly` replicated vdisks can be created with a smaller block size and assigned to the GPFS file system storage pool. The `dataOnly` Reed-Solomon vdisks can be created with a larger block size and assigned to GPFS file system data storage pools. When using multiple storage pools, a GPFS placement policy must be installed to direct file system data to non-system storage pools.

  When write performance optimization is not important, it is acceptable to use Reed-Solomon vdisks as `dataAndMetadata` NSDs for better space efficiency.

- When assigning the failure groups to vdisk NSDs in a GPFS file system, the ESS building block should be considered the common point of failure. All vdisks within all recovery groups in a given ESS building block should be assigned the same failure group number. An exception to this is when the cluster consists of only one ESS building block. In this case, failure groups should be associated with recovery groups rather than with the entire ESS building block.

  Within a recovery group, all file system vdisk NSDs should be assigned the same failure group. If there is more than one ESS building block, all file system vdisk NSDs within both recovery groups of a building block should be assigned the same failure group.

- Attaching storage that is not associated with IBM Storage Scale RAID (SAN-attached disks, for example) to ESS NSD server nodes is *not* supported.

- Because of possible differences in performance and availability characteristics, mixing IBM Storage Scale RAID vdisks with disks that are not associated with IBM Storage Scale RAID in the same storage pool in a file system is *not* recommended. This recommendation applies when mixing different types

of storage in the same storage pool. A good IBM Storage Scale planning practice is to put storage with similar characteristics in the same storage pool.

Appendix B. IBM Storage Scale RAID commands

Descriptions of these IBM Storage Scale RAID commands follow:

For information about other IBM Storage Scale commands, see the *IBM Storage Scale: Command and Programming Reference*.

mmaddcomp command

Adds one or more storage components.

Synopsis

```
mmaddcomp PartNumber
            [--serial-number SerialNumber] [--name Name]
            [--display-id DisplayId] [--replace] [--dry-run]
```

or

```
mmaddcomp -F StanzaFile [--replace] [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmaddcomp` command adds one or more new storage components. A default name is assigned if one is not specified.

Parameters

PartNumber
Specifies the part number or model type of the component to be added.

--serial-number *SerialNumber*
Specifies a serial number to be assigned to the component. If this serial number matches that of an existing component, the command will fail unless `--replace` is also specified.

--name *Name*
Specifies a name to be assigned to the component. A default name is assigned if one is not specified.

--display-id *DisplayID*
Specifies a display ID to be assigned to the component. This applies to storage enclosures only.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for the component definitions in the stanza file is as follows:

```
%comp:
 partNumber=PartNumber
 serialNumber=SerialNumber
 name=Name
 displayId=DisplayId
```

where:

partNumber=*PartNumber*
Specifies the part number or model type of the component to be added. This is a required parameter.

serialNumber=*SerialNumber*
Specifies a serial number to be assigned to the component. If this serial number matches that of an existing component, the command will fail unless `--replace` is also specified.

name=*Name*
Specifies a name to be assigned to the component. A default name is assigned if one is not specified.

displayId=*DisplayId*
Specifies a display ID to be assigned to the component. This applies to storage enclosures only.

--replace
Indicates that the command is to replace an existing component that has a matching serial number or component ID.

--dry-run
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmaddcomp command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To define a rack and see the result, enter the following commands:

```
mmaddcomp 1410HPA --name R01C01
mmlscomp
```

The system displays output similar to this:

```
    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HPA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  1818-80E     SV12345001     1818-80E-SV12345001
      2  1818-80E     SV12345007     1818-80E-SV12345007
      3  1818-80E     SV12345003     1818-80E-SV12345003
      4  1818-80E     SV12345005     1818-80E-SV12345005
      5  1818-80E     SV12345004     1818-80E-SV12345004
      6  1818-80E     SV12345002     1818-80E-SV12345002
      7  1818-80E     SV12345008     1818-80E-SV12345008
      8  1818-80E     SV12345006     1818-80E-SV12345006
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddcompspec command" on page 266
- "mmchcomp command" on page 273

- “mmchcomploc command” on page 275
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmaddcompspec command

Adds one or more new storage component specifications.

Synopsis

```
mmaddcompspec PartNumber CompType
              [--height Height] [--pci-slots PciSlots] [--drive-slots DriveSlots]
              [--vendor-id VendorId] [--product-id ProductId]
              [--has-display-id { yes | no } ] [--description Description]
              [--replace] [--dry-run]
```

or

```
mmaddcompspec -F StanzaFile [--ignore-unknown] [--replace] [--dry-run]
```

or

```
mmaddcompspec default [--replace] [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmaddcompspec` command adds one or more new storage component specifications. A default name is assigned if one is not specified.

Parameters

PartNumber
Specifies the part number or model type of the component to be added.

CompType
Specifies the component type. Valid values are: `rack`, `server`, `storageEnclosure`, and `storageServer`.

--height ***Height***
Specifies the component height in rack units (U).

--pci-slots ***PciSlots***
Specifies the number of PCI slots. This parameter applies to servers only.

--drive-slots ***DriveSlots***
Specifies the number of drive slots. This parameter applies to storage enclosures only.

--vendor-id ***VendorId***
Specifies the vendor ID reported by SCSI inquiry. This parameter applies to storage enclosures only.

--product-id ***ProductId***
Specifies the product ID reported by SCSI inquiry. This parameter applies to storage enclosures only.

--has-display-id yes | no
Indicates whether the component has a programmable display ID. This parameter applies to storage enclosures only.

--description ***Description***
Specifies a brief description of the component.

--replace
Replaces an existing specification with a matching part number.

--dry-run
Runs the command without making any permanent changes.

-F ***StanzaFile***
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for the component definitions in the stanza file is as follows:

```
%compSpec:
 partNumber=PartNumber
 compType=CompType
 height=Height
 description=Description
 vendorId=VendorId
 productId=ProductId
 driveSlots=DriveSlots
 hasDisplayId=HasDisplayId
```

--ignore-unknown
Ignores unknown attributes in compSpec stanzas.

default
Add all of the predefined component specifications.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmaddcompspec` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To define a generic 60U rack and see the result, enter the following commands:

```
mmaddcompspec RACK60U rack --height 60 --description "Generic 60u rack"

mmlscompspec --type rack
```

The system displays output similar to this:

```
    Rack Specifications

Part Number  Height  Description
-----------  ------  -----------------------------------------------------
1410HEA          42  IBM Intelligent Cluster 42U 1200mm Deep Expansion Rack
1410HPA          42  IBM Intelligent Cluster 42U 1200mm Deep Primary Rack
9308RC4          42  IBM 42U Enterprise Rack
RACK42U          42  Generic 42U rack
RACK60U          60  Generic 60u rack
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmchcomp command" on page 273
- "mmchcomploc command" on page 275
- "mmdelcomp command" on page 295
- "mmdelcomploc command" on page 296
- "mmdelcompspec command" on page 298
- "mmdiscovercomp command" on page 304
- "mmlscomp command" on page 308
- "mmlscomploc command" on page 310
- "mmlscompspec command" on page 312
- "mmsyncdisplayid command" on page 340
- Chapter 7, "Configuring components on the Elastic Storage Server ," on page 75.

Location

`/usr/lpp/mmfs/bin`

mmaddpdisk command

Adds a pdisk to an IBM Storage Scale RAID recovery group.

Synopsis

```
mmaddpdisk RecoveryGroupName -F StanzaFile [--replace] [-v {yes | no}]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmaddpdisk` command adds a pdisk to a recovery group.

Note: The GPFS daemon must be running on both the primary and backup servers to run this command.

Parameters

RecoveryGroupName
Specifies the recovery group to which the pdisks are being added.

-F *StanzaFile*
Specifies a file containing pdisk stanzas that identify the pdisks to be added. For more information about pdisk stanzas, see the the following *IBM Storage Scale RAID: Administration* topic: "Pdisk stanza format" on page 55.

--replace
Indicates that any existing pdisk that has the same name as a pdisk listed in the stanza file is to be replaced. (This is an atomic deletion and addition.)

-v {yes | no}
Verifies that specified pdisks do not belong to an existing recovery group. The default is `-v yes`.

Specify `-v no` only when you are certain that the specified disk does not belong to an existing recovery group. Using `-v no` on a disk that already belongs to a recovery group will corrupt that recovery group.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmaddpdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

In this example, suppose the input stanza file, `pdisk.c033d2`, contains the following lines:

```
%pdisk: pdiskName=c033d2
        device=/dev/hdisk674
        da=DA2
        nPathActive=2
        nPathTotal=4
```

This command example shows how to add the pdisk described in stanza file `pdisk.c033d2` to recovery group `000DE37BOT`:

```
mmaddpdisk 000DE37BOT -F pdisk.c033d2
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmchpdisk command" on page 283

- “mmdelpdisk command” on page 299
- “mmlspdisk command” on page 324
- “mmlsrecoverygroup command” on page 327.

Location

`/usr/lpp/mmfs/bin`

mmchcarrier command

Allows IBM Storage Scale RAID pdisks to be physically removed and replaced.

Synopsis

```
mmchcarrier RecoveryGroupName --release
               {[--pdisk "Pdisk[;Pdisk]" [--location "Location[;Location]"]}
               [--force-release] [--force-rg]
```

or

```
mmchcarrier RecoveryGroupName --resume
               {[--pdisk "Pdisk[;Pdisk]" [--location "Location[;Location]"]}
               [--force-rg]
```

or

```
mmchcarrier RecoveryGroupName --replace
               {[--pdisk "Pdisk[;Pdisk]" [--location "Location[;Location]"]}
               [-v {yes|no}] [--force-fru] [--force-rg] [--nsd-version {1|2}]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmchcarrier` command is used to control disk carriers and replace failed pdisks.

Replacing a pdisk requires the following three steps:

1. Run the `mmchcarrier --release` command to prepare the carrier for removal.

   The `mmchcarrier --release` command suspends I/O to all disks in the carrier, turns off power to the disks, illuminates identify lights on the carrier, and unlocks the carrier latch (if applicable).
2. Remove the carrier from the disk drawer, replace the failed disk or disks with new disks, and reinsert the carrier into the disk drawer.
3. Run the `mmchcarrier --replace` command to complete the replacement.

   The `mmchcarrier --replace` command powers on the disks, verifies that the new disks have been installed, resumes I/O, and begins the rebuilding and rebalancing process onto the new disks.

   Note: New disks will take the name of the replaced pdisks. In the event that replaced pdisks have not completely drained, they will be given a temporary name consisting of the old pdisk name with a suffix of the form *#nnnn*. The temporary pdisk will have the `adminDrain` pdisk state flag set and will be deleted once drained. For example, a pdisk named p25 will receive a temporary name similar to `p25#0010` when the `adminDrain` state flag is set. This allows the new disk that is replacing it to be named p25 immediately rather than waiting for the old disk to be completely drained and deleted. Until the draining and deleting process completes, both the new pdisk p25 and the old pdisk `p25#0010` will show up in the output of the `mmlsrecoverygroup` and `mmlspdisk` commands.

Both the release and replace commands require either a recovery group name and a location code, or a recovery group name and a pdisk name to identify the carrier and particular disk slot within the carrier. It is acceptable to provide more than one location code or pdisk name to replace multiple disks within the same carrier.

The `mmchcarrier --resume` command reverses the effect of the release command without doing disk replacements. It can be used to cancel the disk replacement procedure after running the `mmchcarrier --release` command.

Parameters

RecoveryGroupName
Specifies the name of the recovery group to which the carrier belongs. This is used to identify the active server where the low level commands will be issued.

--release
Suspends all disks in the carrier, activates identify lights, and unlocks the carrier.

--resume
Resumes all disks in the carrier without doing disk replacements.

--replace
Formats the replacement disks for use and resumes all disks in the carrier.

--pdisk
Specifies the target pdisk or pdisks and identifies the carrier. All specified pdisks must belong to the same carrier.

--location
Specifies the target pdisk or pdisks and identifies the carrier by location code. All specified pdisks must belong to the same carrier. If this option is used, the `location` code must be obtained from the output of the `mmlspdisk` command. There is a field `location` listed for each pdisk.

--force-release
This is a force flag for the `--release` option, to release the carrier even if the target is not marked for replacement. This command is intended to temporarily release a carrier. It should not be used to force disk replacement. Disks marked for replacement are identified via the `mmlspdisk --replace` command.

--force-fru
This is a force flag for the `--replace` option, to allow the replacement even if the field replaceable unit (FRU) number of the new disk does not match that of the old disk.

--force-rg
This is a force flag for the `--release`, `--resume`, and `--replace` options to allow actions on the carrier even if all the pdisks do not belong to the same recovery group.

--nsd-version
Specifies the desired Nsd version for the replacement disks. The value can be either 1 or 2. This parameter is only effective with recovery group version 4.2.0.1 or up. If the Nsd version for the disks marked for replacement is known, this parameter will be ignored. If the Nsd version for the disk marked for replacement is not known, and if this parameter is not specified, the pdisk Nsd version will be 2 for recovery group version 4.2.0.1 or up. For recovery group version 4.1.0.1 or lower, the Nsd version can only be 1.

-v {yes | no}
Verification flag for the `--replace` option; indicates whether or not to verify that the new disk does not already have a valid pdisk descriptor. The default is `-v yes`.

Specify `-v no` to allow a disk that was formerly part of some other recovery group to be reused.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchcarrier` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. The following command example shows how to release the carrier containing failed pdisk `c014d3` in recovery group `000DE37BOT`:

   ```
   mmchcarrier 000DE37BOT --release --pdisk c014d3
   ```

   The system displays output similar to the following:

   ```
   [I] Suspending pdisk c014d1 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D1.
   [I] Suspending pdisk c014d2 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D2.
   [I] Suspending pdisk c014d3 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D3.
   [I] Suspending pdisk c014d4 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D4.
   [I] Carrier released.

     - Remove carrier.
     - Replace disk in location 78AD.001.000DE37-C14-D3 with FRU 74Y4936.
     - Reinsert carrier.
     - Issue the following command:

         mmchcarrier 000DE37TOP --replace --pdisk 'c014d3'
   ```

2. The following command example shows how to tell IBM Storage Scale that the carrier containing pdisk `c014d3` in recovery group `000DE37BOT` has been reinserted and is ready to be brought back online:

   ```
   mmchcarrier 000DE37BOT --replace --pdisk c014d3
   ```

   The system displays output similar to the following:

   ```
   [I] The following pdisks will be formatted on node server1:
       /dev/rhdisk354
   [I] Pdisk c014d3 of RG 000DE37TOP successfully replaced.
   [I] Resuming pdisk c014d1 of RG 000DE37TOP.
   [I] Resuming pdisk c014d2 of RG 000DE37TOP.
   [I] Resuming pdisk c014d3#162 of RG 000DE37TOP.
   [I] Resuming pdisk c014d4 of RG 000DE37TOP.
   [I] Carrier resumed.
   ```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmchpdisk command" on page 283
- "mmchrecoverygroup command" on page 285.
- "mmlspdisk command" on page 324

Location

`/usr/lpp/mmfs/bin`

mmchcomp command

Changes attributes associated with one or more storage components.

Synopsis

```
mmchcomp Component
            [--part-number PartNumber] [--serial-number SerialNumber]
            [--display-id DisplayId] [--name Name] [--dry-run]
```

or

```
mmchcomp -F StanzaFile [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmchcomp` command changes the various attributes associated with one or more components.

Parameters

Component
Specifies the current name, serial number, or ID of a single component for which one or more attributes are to be changed.

--part-number *PartNumber*
Specifies a new part number to be assigned to the component. You cannot specify a part number that would change the component type.

--serial-number *SerialNumber*
Specifies a new serial number to be assigned to the component.

--display-id *DisplayID*
Specifies a new display ID to be assigned to the component. This applies to storage enclosures only. After setting this attribute, use `mmsyncdisplayid` to update the storage enclosure hardware.

--name *Name*
Specifies a new name to be assigned to the component.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for the component definitions in the stanza file is as follows:

```
%comp:
 compID=Component
 partNumber=PartNumber
 serialNumber=SerialNumber
 name=Name
 displayId=DisplayId
 nodeNumber=NodeNumber
```

where:

compID=*Component*
Specifies the current name, serial number, or ID of a single component for which one or more attributes are to be changed. This is a required parameter.

partNumber=*PartNumber*
Specifies a new part number to be assigned to the component. You cannot specify a part number that would change the component type.

serialNumber=*SerialNumber*
Specifies a new serial number to be assigned to the component.

name=*Name*
Specifies a name to be assigned to the component. A default name is assigned if one is not specified.

displayId=*DisplayId*
Specifies a new display ID to be assigned to the component. This applies to storage enclosures only. After setting this attribute, use `mmsyncdisplayid` to update the storage enclosure hardware.

--dry-run
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchcomp` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To update the name of an enclosure (in this case, the third enclosure) so that the name includes the product serial number, and to see the result, enter the following commands:

```
mmchcomp 3 --name G1E3-SX98760044
mmlscomp --type storageenclosure
```

The system displays output similar to this:

```
    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  1818-80E     SV12345001     1818-80E-SV12345001  21
      2  1818-80E     SV12345007     1818-80E-SV12345007  01
      3  1818-80E     SV12345003     G1E3-SX98760044     13
      4  1818-80E     SV12345005     1818-80E-SV12345005  05
      5  1818-80E     SV12345004     1818-80E-SV12345004  37
      6  1818-80E     SV12345002     1818-80E-SV12345002  25
      7  1818-80E     SV12345008     1818-80E-SV12345008  17
      8  1818-80E     SV12345006     1818-80E-SV12345006  33
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddcomp command" on page 264
- "mmaddcompspec command" on page 266
- "mmchcomploc command" on page 275
- "mmchenclosure command" on page 277

- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmchcomploc command

Changes the location of one or more storage components.

Synopsis

```
mmchcomploc Component Container Position [--dry-run]
```

or

```
mmchcomploc -F StanzaFile [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmchcomploc` command changes the location (container and position within the container) of one or more storage components.

Parameters

Component
Specifies the current name, serial number, or component ID of a single component for which the location is to be changed.

Container
Specifies the name, serial number, or component ID of the container to be assigned to the component.

Position
Specifies an integer value that identifies the position within the container to be assigned to the component.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for the component definitions in the stanza file is as follows:

```
%comp:
 compId=Component
 containerId=Container
 position=Position
```

where:

compId=*Component*
Specifies the current name, serial number, or component ID of a single component for which the location is to be changed. This is a required parameter.

containerId=*Container*
Specifies the name, serial number, or component ID of the container to be assigned to the component. This is a required parameter.

position=*Position*
Specifies an integer value that identifies the position within the container to be assigned to the component. This is a required parameter.

--dry-run
Indicates that the changes that result from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchcomploc` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

To place a component in a rack and see the result, enter the following commands:

```
mmchcomploc SV12345003 R01C01 13
mmlscomploc
```

The system displays output similar to this:

```
Component           Location
------------------  ------------------
1818-80E-SV12345003 Rack R01C01 U13-16
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279

- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmchenclosure command

A service aid to be run before replacing disk enclosure components. Identifies whether it is safe to complete the repair action or whether IBM Storage Scale needs to be shut down first.

Synopsis

```
mmchenclosure SerialNumber
              --component { canister | cpu | currentSensor | dcm | dimm | drawer | enclosure |
esm | expander | fan | powerSupply | sideplane | tempSensor | voltageSensor }
              --component-id Id
```

Availability

Available with the Elastic Storage Server.

Description

Use the `mmchenclosure` command to check whether it is safe to replace a component while the GPFS daemon is active. The command will indicate whether there are issues that would require the GPFS daemon to be stopped before proceeding.

In the case of failures, the field replaceable units are:

- Drives.
- Enclosure drawers. Any failure that is related to a drawer control module (DCM)'s component ID (component type: `dcm`) requires a drawer replacement.
- Environmental service modules (ESMs).
- Expanders.
- Fan assemblies.
- Power supplies.
- Sideplanes.

Parameters

--serial-number *SerialNumber*
Specifies the serial number of the storage enclosure (as identified in the output of the `mmlsenclosure` command).

--component { canister | cpu | currentSensor | dcm | dimm | drawer | enclosure | esm | expander | fan | powerSupply | sideplane | tempSensor | voltageSensor }
Specifies the type of component that needs to be changed. The component types follow:

canister
Is a canister module.

cpu
Is a central processing unit.

currentSensor
Is a current sensor.

dcm
Is a drawer control module.

dimm
Is a memory module.

drawer
Is a drawer.

enclosure
Is a storage enclosure.

esm
Is an environmental service module.

expander
Is a sub-expander.

fan
Is a cooling fan.

powerSupply
Is a power supply.

sideplane
Is an expander board.

tempSensor
Is a temperature sensor.

voltageSensor
Is a voltage sensor.

--component-id *Id*
Specifies the component ID (as identified in the output of the `mmlsenclosure` command).

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchenclosure` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. Before replacing a fan with a component ID of 2_BOT_RIGHT on enclosure SV13306129, run:

```
mmchenclosure SV13306129 --component fan --component-id 2_BOT_RIGHT
```

   In this example, `mmlsenclosure` did not show the fan as needing to be replaced. The system displays information similar to this:

```
mmchenclosure: [E] Storage enclosure SV13306129 component fan component id 2_BOT_RGHT is not failed.
 Service is not required.
Storage enclosure SV13306129 component fan component id 2_BOT_RGHT is not failed. Service is not required.
 mmchenclosure: Command failed.  Examine previous error messages to determine cause.
```

2. Before replacing a fan with a component ID of 1_BOT_LEFT on enclosure SV13306129, run:

```
mmchenclosure SV13306129 --component fan --component-id 1_BOT_LEFT
```

   The system displays output similar to this:

```
mmchenclosure: Proceed with the replace operation.
```

3. Before replacing the drawer associated with DCM component ID DCM_0A on enclosure SV13306129, run:

```
mmchenclosure SV13306129 --component dcm --component-id DCM_0A
```

   The system displays output similar to this:

```
     mmchenclosure: Shutdown GPFS on the following nodes and then proceed
     with the replace operation.
     Shutdown GPFS on the following nodes and then proceed with the replace
     operation.
```

4. Before replacing an ESM with a component ID of ESM_A on enclosure SV13306129, run:

```
mmchenclosure SV13306129 --component esm --component-id ESM_A
```

   The system displays output similar to this:

```
mmchenclosure: Enclosure SV13306129 ESM A is currently redundant.
mmchenclosure: All in-use disks on enclosure SV13306129 ESM A have redundant paths.
mmchenclosure: Proceed with the replace operation.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topic:

- “mmlsenclosure command” on page 314.

See also:

- The `/usr/lpp/mmfs/samples/vdisk/chdrawer` sample script.

Location

`/usr/lpp/mmfs/bin`

mmchfirmware command

Changes the firmware levels on one or more storage components.

Synopsis

```
mmchfirmware --type {storage-enclosure | drive | host-adapter}
             [ --update-id PREVIOUS]
```

```
              [ --serial-number SerialNumber ]
              [ -N { Node[,Node...] | NodeFile | NodeClass } ]
              [--stop-on-error { yes | no } ]
              [ --fast-offline ]
              [ --dry-run ]
              [ -v {yes | no}
```

Availability

Available with the Elastic Storage Server.

Description

Use the **mmchfirmware** command to update the firmware level on one or more storage components.

A particular component is updated only if it satisfies the `--update-id`, `--type`, `--serial-number`, and `-N` options when they are specified.

Consequently, when you update drives on a cluster that has the IBM Storage Scale daemons active, you need to use the `-N` option to include both servers of the ESS building blocks that might be affected.

Note: The firmware on each of the two sides of an enclosure can be upgraded independently. During a serial update procedure, the two sides of an enclosure typically operate with mismatched firmware levels for a short period (a few minutes), during which time the enclosure presents a warning because of the firmware mismatch.

Parameters

--type {storage-enclosure | drive | host-adapter}
Specifies the type of component for which firmware is to be updated.

--update-id PREVIOUS
Only the keyword PREVIOUS is supported. The default is to install the newest available firmware.

The keyword PREVIOUS assumes that you already loaded the latest firmware available and now want to back off to the previous level.

--serial-number *SerialNumber*
Specifies the serial number of a particular component to be updated.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes to be included in the update. This command supports all defined node classes. Only components directly accessible from one of the specified nodes are included in the component selection list. The default is to run on the invoking node.

For general information about how to specify node names, see the following *IBM Storage Scale RAID: Administration* topic: “Specifying nodes as input to IBM Storage Scale RAID commands” on page 18.

--stop-on-error {yes | no}
Specifies whether the command is to stop when a firmware update fails. The default value is `yes`.

--fast-offline
This option is deprecated. It was intended to increase parallelism when updating drive firmware if IBM Storage Scale was shut down on all the nodes in the cluster. Parallelism automatically happens for the following cases:

- **`host-adapter`**: Runs in parallel on all of the nodes that are specified unless debugging is enabled.

  Note: IBM Storage Scale must be shut down on all of the specified nodes.
- **`drive`**: Parallelism automatically happens if IBM Storage Scale is shut down on all of the potentially affected nodes.

  Note: Potentially affected nodes include the input nodes and any other nodes that have access to the same storage enclosures and drives.

- **storage-enclosure**: Parallelism automatically happens if IBM Storage Scale is shut down on all of the potentially affected nodes.

  Note: Potentially affected nodes include the input nodes and any other nodes that have access to the same storage enclosures and drives.

--dry-run
Specifies that the command is to be run without updating any firmware.

-v {yes | no}
Specifies whether the command is to verify that the firmware version is newer than the current version. A value of yes specifies that the firmware is to be updated only if it is a newer version than the firmware already installed on the device. A value of no specifies that the firmware is to be updated in all cases, even if its version number indicates that it is an earlier level or the same level as the firmware already installed on the device. The default value is yes.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the **mmchfirmware** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. To update an enclosure (in this case, enclosure SV13306129) to the latest firmware that was supplied, issue the following command:

```
mmchfirmware --type storage-enclosure --serial-number SV13306129
```

   The system displays output similar to this:

```
Mon Dec  3 08:19:12 EST 2012: mmchfirmware: Processing node c45f01n01-ib0.gpfs.net
c45f01n01-ib0: update-directory /usr/lpp/mmfs/updates/latest/firmware/enclosure/ProductId
c45f01n01-ib0: Found storage-enclosure DCS3700 SV13306129, update-id esm0375.esm.
c45f01n01-ib0: Updating enclosure firmware ESM_A. c45f01n01-ib0: Updating enclosure firmware
ESM_B.
```

2. To update the drive firmware, issue the following command:

```
mmchfirmware --type drive
```

   The system displays output similar to this:

```
Mon Dec  3 09:10:47 EST 2012: mmchfirmware: Processing node c45f02n01-ib0.gpfs.net
c45f02n01-ib0: update-directory /usr/lpp/mmfs/updates/latest/firmware/drive.
c45f02n01-ib0: Found drive firmware update-id ibm_fw_hdd_sas-9.99_linux_32-64_gn
r.zip.
c45f02n01-ib0: Updating firmware for drives in recovery group rg.
c45f02n01-ib0: Updating firmware for drives sg46,sg179,sg183.
c45f02n01-ib0: Updating firmware for drives sg48,sg158,sg171.
c45f02n01-ib0: Updating firmware for drives sg45,sg131,sg352.
c45f02n01-ib0: Updating firmware for drives sg304,sg123.
c45f02n01-ib0: Updating firmware for drives sg193,sg180.
...
c45f02n01-ib0: Updating firmware for drives not contained in a recovery group.
```

```
c45f02n01-ib0: Updating firmware for drives sg201,sg29,sg250,sg297,sg36.
c45f02n01-ib0: Updating firmware for drives sg345,sg97,sg200,sg249,sg199.
c45f02n01-ib0: Updating firmware for drives sg69,sg28,sg298,sg346.
```

3. To update the drive firmware for a specific drive, you first must determine the serial number of the drive. You can discover the drive serial number by using one of two options, as follows:

   a. Issue the following command:

   ```
   mmlspdisk RecoveryGroupName --pdisk
   ```

   The system displays output similar to this:

   ```
   >mmlspdisk BB1RGL --pdisk e4d5s04
   pdisk:
      replacementPriority = 1000
      name = "e4d5s04"
      device = "/dev/sdbf,/dev/sdhy"
      recoveryGroup = "BB1RGL"
      declusteredArray = "DA2"
      .
      .
      .
      hardware = "IBM-ESXS ST2000NM0001 BC4A Z1P4K3VF00009320N6RS"
   ```

   The drive serial number is the last part of the `hardware` value: Z1P4K3VF00009320N6RS in this example.

   b. Issue the following command:

   ```
   mmgetpdisktopology > OutputFile
   ```

   Edit your output file (`out1`, for example) and search for the pdisk name or device path that you want to update:

   ```
   cat out1 | grep e4d5s04
   ```

   The system displays output similar to this:

   ```
     >cat out1 | grep e4d5s04
      sdbf,sdhy:0:/dev/sdbf,/dev/sdhy:C0A82D0B5384A13C|e4d5s04|BB1RGL|C0A82D0B53849EE7|DA2||
   1:
   naa.5000C50055D5F0E7:0000%3a11%3a00.0/7.0.51.0,0000%3a8b%3a00.0/8.0.51.0:[7.0.51.0],
   [8.0.51.0]:/dev/sg59,/dev/sg237:IBM-ESXS:ST2000NM0001:BC4A:Z1P4K3VF00009320N6RS:46W6911:
   2000398934016:naa.500062B200422340,naa.500062B200422280:50080e5245ded03f.50080e5245dec03f:
   SV13306129:0/1:5-4:5000c50055d5f0e5.5000c50055d5f0e6:sg8,sg186:
   ```

   The drive serial number is field 12 of the colon-delimited fields. Again, you see that the drive serial number is Z1P4K3VF00009320N6RS in this example.

   Now issue the following command:

   ```
   mmchfirmware --type drive --serial-number Z1P4K3VF00009320N6RS
   ```

4. This example shows you how to update the drive firmware when the IBM Storage Scale RAID cluster is shut down. Updating the firmware during a maintenance shutdown is much faster than updating the firmware while IBM Storage Scale is active. With IBM Storage Scale active, the parallelism is reduced, pdisks must be suspended and resumed, and declustered arrays must be rebalanced between each update iteration.

   With IBM Storage Scale shut down in the cluster, run this command:

   ```
   mmchfirmware --type drive --fast-offline
   ```

Progress messages are suppressed during this concurrent update because they are interspersed from different nodes. To display progress messages, run this command:

```
DEBUGmmchfirmware=1 mmchfirmware --type drive --fast-offline
```

5. To update the host adapter firmware, run this command:

```
mmchfirmware --type host-adapter
```

Note: The node must be down where the host adapter firmware is being updated.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320.

Location

`/usr/lpp/mmfs/bin`

mmchpdisk command

Changes IBM Storage Scale RAID pdisk states. This command is to be used only in extreme situations under the guidance of IBM service personnel.

Synopsis

```
mmchpdisk RecoveryGroupName --pdisk PdiskName
      { --simulate-dead | --simulate-failing | --kill | --revive | --revive-failing |
     --revive-slow | --diagnose |--suspend | --resume |
     --begin-service-drain | --end-service-drain }
     [ --identify {on|off}] [ --clear-error-counters ]
```

Availability

Available on all IBM Storage Scale editions.

Description

The **mmchpdisk** command changes the states of pdisks.

Attention: This command is to be used only in extreme situations under the guidance of IBM service personnel.

Parameters

RecoveryGroupName
Specifies the recovery group that contains the pdisk for which one or more states are to be changed.

--pdisk *PdiskName*
Specifies the target pdisk.

--simulate-dead
Specifies that the disk is being treated as if it were dead.

Attention:

- This command can cause permanent data loss, so do not use it on production systems.

- This option must be used with caution; if the total number of failures in a declustered array exceeds the fault tolerance of any vdisk in that array, permanent data loss might result.

--simulate-failing
Specifies that the disk is being treated as if it were failing.

Attention: This option must be used with caution; if the total number of failures in a declustered array exceeds the fault tolerance of any vdisk in that array, permanent data loss might result.

--kill
The **--kill** option is deprecated. If it is run, it acts like **--simulate-failing**.

--revive
Attempts to make a failed disk usable again by removing `dead`, `failing`, and `readonly` pdisk state flags. Data can become readable again if the disk was not rebuilt onto spare space; however, any data that was already reported as lost cannot be recovered.

--revive-failing
Clears the "failing" pdisk state flag and resets all of the disk hospital bit error rate history for the pdisk.

Note: It might take several months to rebuild this history. Do not use this option to clear `simulatedFailing`; use **--revive** instead.

--revive-slow
Clears the "slow" pdisk state flag and resets all of the disk hospital I/O performance history for the pdisk.

--diagnose
Runs basic tests on the pdisk. If no problems are found, the pdisk state automatically returns to ok.

--suspend
Suspends I/O to the pdisk until a subsequent resume command is given. If a pdisk remains in the `suspended` state for longer than a predefined timeout period, IBM Storage Scale RAID begins rebuilding the data from that pdisk into spare space.

Attention: Use this option with caution and only when performing maintenance on disks manually, bypassing the automatic system provided by **mmchcarrier**.

If a pdisk is removed by using this option, vdisks that store data on it are temporarily degraded, requiring data that was stored on the removed pdisk to be rebuilt from redundant data. Also, if you try to remove more pdisks than the redundancy level of the least redundant vdisk in that declustered array, data becomes inaccessible.

Therefore, when preparing to remove a pdisk, use the **--begin-service-drain** and **--end-service-drain** options instead of this option.

--resume
Cancels a previously run **mmchpdisk --suspend** command and resumes use of the pdisk.

Use this option only when performing maintenance on disks manually and bypassing the automatic system provided by **mmchcarrier**.

--begin-service-drain
Starts draining the specified pdisk so that it can be temporarily removed. After issuing the command with this option, wait until the pdisk is drained before removing it.

Note: This process requires sufficient spare space in the declustered array for the data that is to be drained. If the available spare space is insufficient, it can be increased with the **mmchrecoverygroup** command.

--end-service-drain
Returns drained data to a pdisk after the pdisk is brought back online.

--identify {on | off}
Turns on or off the disk identify light, if available.

--clear-error-counters
Resets the IOErrors, IOTimeouts, mediaErrors, checksumErrors, and pathErrors counters.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the **mmchpdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

The following command example shows how to instruct IBM Storage Scale to try to revive the failed pdisk c036d3 in recovery group 000DE37B0T:

```
mmchpdisk 000DE37B0T --pdisk c036d3 --revive
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddpdisk command" on page 268
- "mmchcarrier command" on page 270
- "mmdelpdisk command" on page 299
- "mmlspdisk command" on page 324
- "mmlsrecoverygroup command" on page 327.

Location

/usr/lpp/mmfs/bin

mmchrecoverygroup command

Changes IBM Storage Scale RAID recovery group and declustered array attributes.

Synopsis

```
mmchrecoverygroup RecoveryGroupName {--declustered-array DeclusteredArrayName
                                     {[--spares NumberOfSpares]
                                     [--vcd-spares NumberOfVCDSpares]
                                     [--scrub-duration NumberOfDays]
                                     [--replace-threshold NumberOfDisks]
                                     [--trim-device {Auto | No}]}
                                     [--bit-error-rate {enable | disable}]
                                     [--upgrade-nsd-v2]
                                     [--version {VersionString | LATEST}]}
```

or

```
mmchrecoverygroup RecoveryGroupName --active ServerName
```

or

```
mmchrecoverygroup RecoveryGroupName --servers Primary[,Backup] [-v {yes | no}]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmchrecoverygroup` command changes recovery group and declustered array attributes.

`--version` is the only option that applies to the whole recovery group. All other options apply to a declustered array and must be used in conjunction with the `--declustered-array` option.

Parameters

RecoveryGroupName
Specifies the name of the recovery group being changed.

--declustered-array *DeclusteredArrayName*
Specifies the name of the declustered array being changed.

--spares *NumberOfSpares*
Specifies the number of disks' worth of spare space to set aside in the declustered array. This space is used to rebuild the declustered arrays when physical disks fail.

--vcd-spares *NumberOfVCDSpares*
Specifies the number of disks that can be unavailable while full replication of vdisk configuration data (VCD) is still maintained.

--scrub-duration *NumberOfDays*
Specifies the number of days, from 1 to 365, for the duration of the scrub. The default value is 14.

--replace-threshold *NumberOfDisks*
Specifies the number of pdisks that must fail in the declustered array before `mmlsrecoverygroup` will report that service (disk replacement) is needed.

--trim-device
Specifies whether trim to device shall be enabled for this declustered array. **Auto** enables trim to device and **No** disables trim to device.

Note: It is highly recommended to configure trim to device using the **mmvdisk** command.

--bit-error-rate {enable | disable}
Specifies whether bit-error-rate enforcement shall be enabled for this declustered array. If the bit error rate of a disk within a DA is predicted to be beyond an expected threshold, that pdisk will be marked as failing. The default setting for declustered arrays that support bit-error-rate enforcement is `enable`.

--version {*VersionString* | LATEST}
Specifies the recovery group version.

The valid version strings are:

LATEST
Denotes the latest supported version. New recovery groups will default to the latest version.

4.2.0-1
Denotes the version with all features supported by Elastic Storage Server (ESS) 4.0.

4.1.0.1
Denotes the version with all features supported by GPFS Storage Server (GSS) 2.0.

3.5.0.13
Denotes the version with all features supported by GSS 1.5.

3.5.0.5
Denotes the version with all features supported prior to GSS 1.5.

--upgrade-nsd-v2
Changes pdisks' NSD version to version 2. This option is enabled when the recovery group version is 4.2.0.1.

--active *ServerName*
Changes the active server for the recovery group.

--servers *Primary*[, *Backup*]
Changes the defined list of recovery group servers.

Note: To use this option, all file systems that use this recovery group must be unmounted if the primary and backup servers are not just being swapped.

-v {yes | no}
Specifies whether the new server or servers should verify access to the pdisks of the recovery group. The default is `-v yes`. Use `-v no` to specify that configuration changes should be made without verifying pdisk access; for example, you could use this if all the servers were down.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchrecoverygroup` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. The following command example shows how to change the number of spares to one and the replacement threshold to one for declustered array DA4 in recovery group `000DE37TOP`:

```
mmchrecoverygroup 000DE37TOP --declustered-array DA4 --spares 1 --replace-threshold 1
```

2. The following command example shows how to change the scrub duration to one day for declustered array DA2 of recovery group `000DE37BOT`:

```
mmchrecoverygroup 000DE37BOT --declustered-array DA2 --scrub-duration 1
```

3. The following command example shows how to replace the servers for a recovery group. In this example, assume that the two current servers for recovery group RG1 have been taken down for extended maintenance. IBM Storage Scale is shut down on these current RG1 servers. The new servers have already been configured to be recovery group servers, with `mmchconfig` parameter `nsdRAIDTracks` set to a nonzero value. The disks for RG1 have not yet been connected to the new servers, so verification of disk availability must be disabled by specifying `-v no` as shown here:

```
mmchrecoverygroup RG1 --servers newprimary,newbackup -v no
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmchpdisk command” on page 283
- “mmdelrecoverygroup command” on page 301
- “mmcrrecoverygroup command” on page 288
- “mmlsrecoverygroup command” on page 327.

Location

`/usr/lpp/mmfs/bin`

mmcrrecoverygroup command

Creates an IBM Storage Scale RAID recovery group and its component declustered arrays and pdisks and specifies the servers.

Synopsis

```
mmcrrecoverygroup RecoveryGroupName -F StanzaFile --servers {Primary[,Backup]}
                    [--version {VersionString | LATEST}]
                    [-v {yes | no}]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmcrrecoverygroup` command is used to define a cluster-wide recovery group for use by IBM Storage Scale RAID. A recovery group is a set of physical disks shared by up to two server nodes. The set of disks must be partitioned into one or more declustered arrays.

See the following *IBM Storage Scale RAID: Administration* topic: Chapter 4, “Managing IBM Storage Scale RAID,” on page 53.

The pdisk stanzas assigned to a recovery group must contain at least one declustered array that meets the definition of *large*.

While the `mmcrrecoverygroup` command creates the declustered arrays and pdisks and defines the servers, further processing with the `mmcrvdisk` and `mmcrnsd` commands is necessary to create IBM Storage Scale RAID vdisk NSDs within the recovery group.

Note: The recovery group servers must be active to run this command.

Parameters

RecoveryGroupName
Name of the recovery group being created.

-F *StanzaFile*
Specifies a file that includes pdisk stanzas and declustered array stanzas that are used to create the recovery group. The declustered array stanzas are optional. For more information about pdisk stanzas, see the following *IBM Storage Scale RAID: Administration* topic: “Pdisk stanza format” on page 55.

Declustered array stanzas look like the following:

```
%da: daName=DeclusteredArrayName
  spares=Number
  vcdSpares=Number
  replaceThreshold=Number
  scrubDuration=Number
  auLogSize=Number
  nspdEnable={yes|no}
  trimDevice={Auto|No}
```

where:

daName=*DeclusteredArrayName*
Specifies the name of the declustered array for which you are overriding the default values.

spares=*Number*
Specifies the number of disks' worth of spare space to set aside in the declustered array. The number of spares can be 1 or higher. The default values are the following:

1
for arrays with 9 or fewer disks

2
for arrays with 10 or more disks

vcdSpares=*Number*
Specifies the number of disks that can be unavailable while full replication of vdisk configuration data (VCD) is still maintained. The default value is for this value to be the same as the number of spares.

replaceThreshold=*Number*
Specifies the number of pdisks that must fail in the declustered array before `mmlspdisk` will report that pdisks need to be replaced. The default is equal to the number of spares.

scrubDuration=*Number*
Specifies the length of time (in days) by which the scrubbing of entire array must be completed. Valid values are 1 to 60. The default value is 14 days.

auLogSize
Specifies the size of the atomic update log. This value must be chosen very carefully and typically only needs to be set for a declustered array consisting of non-volatile RAM disks. See the following *IBM Storage Scale RAID: Administration* topic: "Configuring IBM Storage Scale RAID recovery groups on the ESS: a sample scenario" on page 115.

nspdEnable {yes|no}
Specifies whether this declustered array should be enabled for network shared pdisks. Declustered arrays that contain non-volatile RAM disks must set `nspdEnable=yes`. The default value is no. See the following *IBM Storage Scale RAID: Administration* topic: "Configuring IBM Storage Scale RAID recovery groups on the ESS: a sample scenario" on page 115.

trimDevice {Auto|No}
Specifies whether trim to device shall be enabled for this declustered array. **Auto** enables trim to device and **No** disables trim to device.

Note: It is highly recommended to configure trim to device using the **mmvdisk** command.

--servers {*Primary*[,** *Backup*]}**
Specifies the primary server and, optionally, a backup server.

--version {*VersionString* | LATEST}
Specifies the recovery group version.

The valid version strings are:

LATEST
Denotes the latest supported version. New recovery groups will default to the latest version.

4.2.0-1
Denotes the version with all features supported by Elastic Storage Server (ESS) 4.0.

4.1.0.1
Denotes the version with all features supported by GPFS Storage Server (GSS) 2.0.

3.5.0.13
Denotes the version with all features supported by GSS 1.5.

3.5.0.5
Denotes the version with all features supported prior to GSS 1.5.

-v {yes | no}
Verification flag that specifies whether each pdisk in the stanza file should only be created if it has not been previously formatted as a pdisk (or NSD). The default is `-v yes`. Use `-v no` to specify that the disks should be created regardless of whether they have been previously formatted or not.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrrecoverygroup` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

Suppose input stanza file `000DE37BOT` contains the following lines:

```
%pdisk: pdiskName=c034d1
        device=/dev/hdisk316
        da=DA1
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c034d2
        device=/dev/hdisk317
        da=DA2
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c034d3
        device=/dev/hdisk318
        da=DA3
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c034d4
        device=/dev/hdisk319
        da=DA4
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c033d1
        device=/dev/hdisk312
        da=LOG
        nPathActive=2
        nPathTotal=4
        nsdformatversion=1
[...]
```

The following command example shows how to create recovery group `000DE37BOT` using stanza file `000DE37BOT`, with `c250f10c08ap01-hf0` as the primary server and `c250f10c07ap01-hf0` as the backup server:

```
mmcrrecoverygroup 000DE37BOT -F 000DE37BOT --servers c250f10c08ap01-hf0,c250f10c07ap01-hf0
```

The system displays output similar to the following:

```
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmchrecoverygroup command” on page 285
- “mmcrvdisk command” on page 291
- “mmdelrecoverygroup command” on page 301
- “mmlsrecoverygroup command” on page 327.

Location

`/usr/lpp/mmfs/bin`

mmcrvdisk command

Creates a vdisk within a declustered array of an IBM Storage Scale RAID recovery group.

Synopsis

```
mmcrvdisk -F StanzaFile [ -v { yes | no } ]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmcrvdisk` command creates one or more vdisks. Upon successful completion of the `mmcrvdisk` command, the vdisk stanza file is rewritten in a form that can be used as input to the `mmcrnsd` command.

The first vdisk that is created in a recovery group must be a log vdisk, which is indicated by a disk usage of `vdiskLog`.

Note: The recovery group must be active to run this command.

Parameters

-F *StanzaFile*
Specifies a file that includes vdisk stanzas identifying the vdisks to be created.

Vdisk stanzas look like the following:

```
%vdisk: vdiskName=VdiskName
        rg=RecoveryGroupName
        da=DeclusteredArrayName
        blocksize=BlockSize
        size=Size
        raidCode=RAIDCode
        diskUsage=DiskUsage
        failureGroup=FailureGroup
        pool=StoragePool
        longTermEventLogSize=LongTermEventLogSize
        shortTermEventLogSize=ShortTermEventLogSize
        fastWriteLogPct=fastWriteLogPct
```

where:

vdiskName=*VdiskName*
Specifies the name you want to assign to the vdisk NSD that is to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string **gpfs**.

Note: This name can contain the following characters only:

Uppercase letters **A** through **Z**
Lowercase letters **a** through **z**
Numerals **0** through **9**
Underscore character (_)

All other characters are not valid.

rg=*RecoveryGroupName*
Specifies the recovery group where the vdisk is to be created.

da=*DeclusteredArrayName*
Specifies the declustered array within the recovery group where the vdisk is to be created.

blocksize=*BlockSize*
Specifies the size of the data blocks for the vdisk. This must match the block size planned for the file system. The valid values are:

For nWayReplicated RAID codes:
256 KiB, 512 KiB, 1 MiB, 2 MiB

For 8+2p and 8+3p RAID codes:
512 KiB, 1 MiB, 2 MiB, 4 MiB, 8 MiB, 16 MiB

Specify this value with the character `K` or `M` (for example, 512K).

If you are using the system pool as metadata only and placing your data in a separate pool, you can specify your metadata-only vdisks with one block size and your data vdisks with a different block size. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool.

size=*Size*
Specifies the size of the vdisk. If `size=`*Size* is omitted, it defaults to using all the available space in the declustered array. The requested vdisk size is equally allocated across all of the pdisks within the declustered array.

raidCode=*RAIDCode*
Specifies the RAID code to be used for the vdisk. Valid codes are the following:

Unreplicated
Indicates no replication. This should only be used for a log tip backup vdisk.

2WayReplication
Indicates two-way replication. This should only be used for a log tip vdisk.

3WayReplication
Indicates three-way replication.

4WayReplication
Indicates four-way replication.

8+2p
Indicates Reed-Solomon 8 + 2p.

8+3p
Indicates Reed-Solomon 8 + 3p.

diskUsage=*DiskUsage*
Specifies a disk usage or accepts the default. With the exception of the `vdiskLog` value, this field is ignored by the `mmcrvdisk` command and is passed unchanged to the output stanza file.

Possible values are the following:

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see the section called "Synchronous mirroring utilizing GPFS replication" in the chapter on "Establishing disaster recovery for your GPFS cluster" of the *IBM Storage Scale: : Administration Guide*.

vdiskLog
Indicates that this is the log home vdisk for the recovery group.

The log home vdisk must be created before any vdisks that are exposed through NSD.

vdiskLogTip
Indicates that this is the log tip vdisk for the recovery group.

vdiskLogTipBackup
Indicates that this is the log tip backup vdisk for the recovery group.

vdiskLogReserved
Indicates a vdisk that should have the same size and RAID code as the log home vdisk but is otherwise unused.

This is used to equalize the space consumption in the declustered arrays that do not contain the log home vdisk.

failureGroup=*FailureGroup*
This field is ignored by the `mmcrvdisk` command and is passed unchanged to the output stanza file. It identifies the failure group to which the vdisk NSD belongs.

pool=*StoragePool*
This field is ignored by the `mmcrvdisk` command and is passed unchanged to the output stanza file. It specifies the name of the storage pool to which the vdisk NSD is assigned.

longTermEventLogSize=*LongTermEventLogSize*
Specifies the size of the long-term event log, rounding up to the block size if not aligned. The default value is 4MB.

`longTermEventLogSize` applies only to the log vdisk.

shortTermEventLogSize=*ShortTermEventLogSize*
Specifies the size of the short-term event log, rounding up to the block size if not aligned. The default value is 1MB.

`shortTermEventLogSize` applies only to the log vdisk.

fastWriteLogPct=*FastWriteLogPct*
Specifies the fraction of the remaining log space to be used for the fast-write log, after allocating space for the event logs. The default value is 90%.

`fastWriteLogPct` applies only to the log vdisk.

-v {yes | no }
Verification flag; a value of `yes` specifies that vdisk creation will only start if the buffer space requirements of other recovery groups can be verified. The default value is `yes`.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrvdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

Assume that input stanza file `000DE37TOP.vdisk` contains the following lines:

```
%vdisk: vdiskName=000DE37TOPLOG
        rg=000DE37TOP
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
        longTermEventLogSize=4M
        shortTermEventLogSize=1M
        fastWriteLogPct=90
%vdisk: vdiskName=000DE37TOPDA1META
        rg=000DE37TOP
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=37
        pool=system
%vdisk: vdiskName=000DE37TOPDA1DATA
        rg=000DE37TOP
        da=DA1
        blocksize=8m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=37
        pool=data
[...]
```

The following command example shows how to create the vdisks described in the stanza file `000DE37TOP.vdisk`:

```
mmcrvdisk -F 000DE37TOP.vdisk
```

The system displays output similar to the following:

```
mmcrvdisk: [I] Processing vdisk 000DE37TOPLOG
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA1META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA1DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA2META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA2DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA3META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA3DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA4META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA4DATA
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmcrrecoverygroup command" on page 288
- "mmdelvdisk command" on page 302
- "mmlsvdisk command" on page 337

Location

`/usr/lpp/mmfs/bin`

mmdelcomp command

Deletes one or more storage components.

Synopsis

```
mmdelcomp Component [--dry-run]
```

or

```
mmdelcomp -F StanzaFile [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmdelcomp` command deletes one or more storage components.

Parameters

Component
Specifies the name, serial number, or component ID of a single component to be deleted.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for each stanza is shown here. Each stanza must include at least one of the parameters listed. If more than one parameter is included in a stanza, the parameters must match the same component.

```
%comp:
 compId=Component
 serialNumber=SerialNumber
 name=Name
```

where:

compId=*Component*
Specifies the component ID of a single component to be deleted.

serialNumber=*SerialNumber*
Specifies the serial number of a single component to be deleted.

name=*Name*
Specifies the name of a single component to be deleted.

--dry-run
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelcomp` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddcomp command" on page 264
- "mmaddcompspec command" on page 266
- "mmchcomp command" on page 273
- "mmchcomploc command" on page 275
- "mmchenclosure command" on page 277
- "mmchfirmware command" on page 279
- "mmdelcomploc command" on page 296
- "mmdelcompspec command" on page 298
- "mmdiscovercomp command" on page 304
- "mmlscomp command" on page 308
- "mmlscomploc command" on page 310
- "mmlscompspec command" on page 312
- "mmlsenclosure command" on page 314
- "mmlsfirmware command" on page 320
- "mmsyncdisplayid command" on page 340
- Chapter 7, "Configuring components on the Elastic Storage Server ," on page 75.

Location

`/usr/lpp/mmfs/bin`

mmdelcomploc command

Deletes one or more storage components from their locations.

Synopsis

```
mmdelcomploc Component Container [--dry-run]
```

or

```
mmdelcomploc -F StanzaFile [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmdelcomploc` command deletes one or more storage components from their locations.

Parameters

Component
Specifies the name, serial number, or component ID of a single component to be deleted from its location.

Container
Specifies the name, serial number, or component ID of the container that holds the component.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

--dry-run
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelcomploc` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddcomp command" on page 264
- "mmaddcompspec command" on page 266
- "mmchcomp command" on page 273
- "mmchcomploc command" on page 275
- "mmchenclosure command" on page 277
- "mmchfirmware command" on page 279
- "mmdelcomp command" on page 295
- "mmdelcompspec command" on page 298
- "mmdiscovercomp command" on page 304
- "mmlscomp command" on page 308
- "mmlscomploc command" on page 310
- "mmlscompspec command" on page 312
- "mmlsenclosure command" on page 314
- "mmlsfirmware command" on page 320
- "mmsyncdisplayid command" on page 340.

Location

`/usr/lpp/mmfs/bin`

mmdelcompspec command

Deletes one or more storage component specifications.

Synopsis

```
mmdelcompspec PartNumber [--dry-run]
```

or

```
mmdelcompspec -F StanzaFile [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmdelcompspec` command deletes one or more storage component specifications.

Parameters

PartNumber
Specifies the part number or model type of the component to be deleted.

-F *StanzaFile*
Specifies the name of a stanza file from which input values are to be read. The stanza file can include stanzas for multiple components.

The correct format for each stanza is shown here. Each stanza must include at least one of the parameters listed. If more than one parameter is included in a stanza, the parameters must match the same component.

```
%compSpec:
 partNumber=PartNumber
```

--dry-run
Runs the command without making any permanent changes.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelcompspec` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

To delete a generic 60U rack, which had been added previously with a part number of RACK60U, enter the following command:

```
mmdelcompspec RACK60U
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmdelpdisk command

Deletes IBM Storage Scale RAID pdisks.

Synopsis

```
mmdelpdisk RecoveryGroupName {--pdisk "PdiskName[;PdiskName...]" | -F StanzaFile} [-a]
```

or

```
mmdelpdisk RecoveryGroupName --declustered-array DeclusteredArrayName
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmdelpdisk` command deletes one or more pdisks. Deleting a pdisk causes any data allocated to that disk to be moved or *rebuilt* (drained) to spare space in the declustered array.

The `mmdelpdisk` command first renames each pdisk that is to be deleted, giving it a temporary name. The command then drains each renamed pdisk to remove all data from it. Finally, the command destroys each renamed pdisk once all data has been drained from it.

Note: The temporary name is obtained by appending a suffix in the form *#nnnn* to the pdisk name. For example, a pdisk named p25 will receive a temporary name similar to p25#010; this allows you to use the `mmaddpdisk` command to add a new pdisk with the name p25 immediately rather than waiting for the old

disk to be completely drained and removed. Until the draining and removing process is complete, both the new pdisk p25 and the old pdisk p25#0010 will show up in the output of the `mmlsrecoverygroup` and `mmlspdisk` commands.

If `mmdelpdisk` is interrupted (by an interrupt signal or by GPFS server failover), the deletion will proceed and will be completed as soon as another IBM Storage Scale RAID server becomes the active vdisk server of the recovery group.

If you wish to delete a declustered array and all pdisks in that declustered array, use the `--declustered-array` *DeclusteredArrayName* form of the command.

The `mmdelpdisk` command cannot be used if the declustered array does not have enough spare space to hold the data that needs to be drained, or if it attempts to reduce the size of a large declustered array below the limit for large declustered arrays. Normally, all of the space in a declustered array is allocated to vdisks and spares, and therefore the only times the `mmdelpdisk` command typically can be used is after adding pdisks, after deleting vdisks, or after reducing the designated number of spares. See the following topics: "mmaddpdisk command" on page 268 and "mmchcarrier command" on page 270.

Note: The recovery group must be active to run this command.

Parameters

RecoveryGroupName
Specifies the recovery group from which the pdisks are being deleted.

--pdisk "*PdiskName*[;*PdiskName*...]"
Specifies a semicolon-separated list of pdisk names identifying the pdisks to be deleted.

-F *StanzaFile*
Specifies a file that contains pdisk stanzas identifying the pdisks to be deleted. For more information about pdisk stanzas, see the the following *IBM Storage Scale RAID: Administration* topic: "Pdisk stanza format" on page 55.

Note: Only the **pdiskName** parameter is used. All other parameters are ignored.

-a
Indicates that the data on the deleted pdisks is to be drained asynchronously. The pdisk will continue to exist, with its name changed to a temporary name, while the deletion progresses.

--declustered-array *DeclusteredArrayName*
Specifies the name of the declustered array whose pdisks are to be deleted.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelpdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

The following command example shows how to remove pdisk `c016d1` from recovery group `000DE37TOP` and have it be drained in the background, thereby returning the administrator immediately to the command prompt:

```
mmdelpdisk 000DE37TOP --pdisk c016d1 -a
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddpdisk command” on page 268
- “mmchpdisk command” on page 283
- “mmdelrecoverygroup command” on page 301
- “mmdelvdisk command” on page 302
- “mmlspdisk command” on page 324
- “mmlsrecoverygroup command” on page 327.

Location

`/usr/lpp/mmfs/bin`

mmdelrecoverygroup command

Deletes an IBM Storage Scale RAID recovery group.

Synopsis

```
mmdelrecoverygroup RecoveryGroupName [-p]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmdelrecoverygroup` command deletes the specified recovery group and the declustered arrays and pdisks that it contains. The recovery group must not contain any vdisks; use the `mmdelvdisk` command to delete vdisks prior to running this command.

Note: The recovery group must be active to run this command, unless the `-p` option is specified.

Parameters

RecoveryGroupName
Specifies the name of the recovery group to delete.

-p
Indicates that the recovery group is permanently damaged and that the recovery group information should be removed from the GPFS cluster configuration data. The `-p` option may be used when the GPFS daemon is down on the recovery group servers.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelrecoverygroup` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

The following command example shows how to delete recovery group `000DE37BOT`:

```
mmdelrecoverygroup 000DE37BOT
```

The system displays output similar to the following:

```
mmdelrecoverygroup: [I] Recovery group 000DE37BOT deleted on node c250f10c08ap01-
hf0.ppd.pok.ibm.com.
mmdelrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmchrecoverygroup command" on page 285
- "mmcrrecoverygroup command" on page 288
- "mmlsrecoverygroup command" on page 327.

Location

`/usr/lpp/mmfs/bin`

mmdelvdisk command

Deletes vdisks from a declustered array in an IBM Storage Scale RAID recovery group.

Synopsis

```
mmdelvdisk {"VdiskName[;VdiskName...]" | -F StanzaFile} [-p | --recovery-group
RecoveryGroupName]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmdelvdisk` command is used to delete vdisks from the declustered arrays of recovery groups. There must be no NSD defined on the specified vdisk; if necessary, use the `mmdelnsd` command prior to running this command.

The log home vdisk in a recovery group cannot be deleted until all the other vdisks in the recovery group have been deleted. Log tip and log tip backup vdisks may be created and deleted at any time, even while work is active on those or other vdisks.

Parameters

***VdiskName*[;*VdiskName*...]**
Specifies the vdisks to be deleted.

-F *StanzaFile*
Specifies the name of a stanza file in which stanzas of the type `%vdisk` identify the vdisks to be deleted. Only the vdisk name is required to be included in the vdisk stanza; however, for a complete description of vdisk stanzas, see the following topic: "mmcrvdisk command" on page 291.

-p
Indicates that the recovery group is permanently damaged, and therefore the vdisk information should be removed from the GPFS cluster configuration data. This option can be used when the GPFS daemon is down on the recovery group servers.

--recovery-group *RecoveryGroupName*
Specifies the name of the recovery group that contains the vdisks. This option is for the rare case where the vdisks have been removed from the GPFS cluster configuration data but are still present in the recovery group.

You can see the vdisks in the recovery group by issuing either of the following commands:

```
mmlsvdisk --recovery-group RecoveryGroupName
```

or

```
mmlsrecoverygroup RecoveryGroupName -L
```

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelvdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

The following command example shows how to delete vdisk `000DE37BOTDA4DATA`:

```
mmdelvdisk 000DE37BOTDA4DATA
```

The system displays output similar to the following:

```
mmdelvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmcrvdisk command" on page 291
- "mmdelpdisk command" on page 299

- “mmdelrecoverygroup command” on page 301
- “mmlsvdisk command” on page 337.

Location

`/usr/lpp/mmfs/bin`

mmdiscovercomp command

Discovers components and adds them to the GPFS cluster configuration.

Synopsis

```
mmdiscovercomp -N {Node[,Node...] | NodeFile | NodeClass} [--dry-run]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmdiscovercomp` command discovers components and adds them to the configuration.

Parameters

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes to which the command applies.

--dry-run
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdiscovercomp` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

To find supported storage enclosures attached to the nodes of a cluster and add them to the configuration, enter the following command:

```
mmdiscovercomp all
```

The system displays output similar to this:

```
Adding enclosure: serial number = SV12345001; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345007; vendor ID = IBM; product ID = DCS3700
```

```
Adding enclosure: serial number = SV12345003; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345005; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345004; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345002; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345008; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345006; vendor ID = IBM; product ID = DCS3700

Storage Enclosures

Status  Comp ID  Component Type    Part Number  Serial Number  Name                Display ID
------  -------  ----------------  -----------  -------------  ------------------  ----------
new           1  storageEnclosure  1818-80E     SV12345001     1818-80E-SV12345001  00
new           2  storageEnclosure  1818-80E     SV12345007     1818-80E-SV12345007  00
new           3  storageEnclosure  1818-80E     SV12345003     1818-80E-SV12345003  00
new           4  storageEnclosure  1818-80E     SV12345005     1818-80E-SV12345005  00
new           5  storageEnclosure  1818-80E     SV12345004     1818-80E-SV12345004  00
new           6  storageEnclosure  1818-80E     SV12345002     1818-80E-SV12345002  00
new           7  storageEnclosure  1818-80E     SV12345008     1818-80E-SV12345008  00
new           8  storageEnclosure  1818-80E     SV12345006     1818-80E-SV12345006  00
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmgetpdisktopology command

Obtains the disk subsystem configuration on an IBM Storage Scale RAID server.

Synopsis

```
mmgetpdisktopology
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmgetpdisktopology` command prints a colon-delimited database of the elements of the disk subsystem on a IBM Storage Scale RAID server. The information is obtained through operating system and device queries. The output is intended to be captured to a file for use by such IBM Storage Scale RAID configuration scripts as **topsummary**, **topselect**, and **mkrginput** in verifying the disk subsystem and creating IBM Storage Scale RAID recovery groups.

The *topology file* that is produced by `mmgetpdisktopology` might also be useful to IBM Service personnel and in diagnosing disk subsystem configuration problems.

IBM Storage Scale does not need to be running to acquire a topology file.

For the purpose of verifying the disk subsystem topology on a server, IBM Storage Scale needs merely to be installed to acquire a topology file.

For the purpose of creating IBM Storage Scale RAID recovery groups, the node where the topology file is acquired needs to be a configured member of a GPFS cluster when the topology is acquired.

It is sometimes useful to peer inside the topology file, even though it is usually considered an opaque object with the primary purpose of facilitating IBM Storage Scale RAID recovery group creation.

Within the topology file are lines of four types: disk, expander, adapter, and host. There is one host line, for the server that this topology describes. There is one line that describes the attributes and relationships for each disk, expander, and adapter present on the server at the time the topology was acquired.

Note: The topology file produced by the **mmgetpdisktopology** command will not include disks that have partition or file system labels and appear to be in use by the operating system or some application (for example, the server's boot disk will never be included). This is to prevent IBM Storage Scale RAID from overwriting user data. If the disk `/dev/deviceName` is expected to be seen in the topology file but is not, make sure that the **`lsblk -ino NAME,TYPE,FSTYPE,MOUNTPOINT /dev/deviceName`** command does not report a possibly stale FSTYPE or MOUNTPOINT value.

Each line in the topology file, regardless of type, contains 21 colon-delimited fields.

The host line is the simplest and is used mainly to record the server hostname and GPFS node name.

The disk, expander, and adapter lines capture information in all 21 fields.

Table 45. Topology file fields

Field	Name	Description
1	`name`	Name for this device
2	`type`	SCSI type (0 for disk, 13 for expander, 31 for adapter)
3	`device`	**/dev** device
4	`nsdid`	GPFS NSD label information for disk devices
5	`wwid`	Unique world-wide identifier
6	`location`	Physical location code
7	`hctl`	Hostbus:controller:target:lun
8	`altdev`	Other **/dev** device
9	`vendor`	Vendor SCSI query
10	`product`	Product name SCSI query
11	`firmware`	Firmware level SCSI query
12	`serial`	Serial number SCSI query
13	`fru`	FRU SCSI query

Table 45. Topology file fields (continued)		
Field	**Name**	**Description**
14	`size`	Size in bytes for disk devices
15	`adapters`	WWIDs of adapters that see this device
16	`expanders`	WWIDs of expanders that see this device
17	`enclosure`	Enclosure that contains this device
18	`port`	Enclosure ESM port that connects the device
19	`slot`	Enclosure slot where the disk resides
20	`sasaddrs`	SAS addresses for the device
21	`sesdev`	Device names of expanders that see the device

Some fields, such as `size`, apply to disks only. Some, such as `enclosure`, do not apply to adapters. Some fields contain two comma-separated values; these are for multiple paths to the same disk or expander. In the case of multiple paths, the order of the comma-separated values is preserved across fields. If **"/dev/sdabc,/dev/sdxyz"** is the `device` field and **"sg57,sg195"** is the `sesdev` field, **sg57** is the expander through which **/dev/sdabc** is connected and **sg195** is the expander through which **/dev/sdxyz** is connected.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmgetpdisktopology` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Example

The following command example shows how to acquire the topology file on a IBM Storage Scale RAID server:

```
mmgetpdisktopology > server1.top
```

The **server1.top** topology file can now be used with the **topsummary**, **topselect**, and **mkrginput** commands.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmcrrecoverygroup command” on page 288
- “mkrginput script” on page 409
- “topselect script” on page 412
- “topsummary script” on page 415.

Location

`/usr/lpp/mmfs/bin`

mmlscomp command

Displays storage components.

Synopsis

```
mmlscomp [--type CompType] [--part-number PartNumber]
           [--serial-number SerialNumber] [--name Name] [--comp-id CompId]
           [--format Format] [--output-file OutputFile] [--sortSortKey]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmlscomp` command displays a list of storage components with attributes that match the specified parameters.

Parameters

--type *CompType*
Specifies a component type, which may include wildcard characters. The resulting list includes those components whose component types (`rack`, `server`, or `storageEnclosure`) match the specified value.

--part-number *PartNumber*
Specifies a part number, which may include wildcard characters. The resulting list includes those components whose part numbers match the specified value.

--serial-number *SerialNumber*
Specifies a serial number, which may include wildcard characters. The resulting list includes those components whose serial numbers match the specified value.

--name *Name*
Specifies a name, which may include wildcard characters. The resulting list includes those components whose names match the specified value.

--comp-id *CompId*
Specifies a component ID, which may include wildcard characters. The resulting list includes those components whose component IDs match the specified value.

--format *Format*
Specifies the format in which the component data will be listed. Valid values are: `colon`, `column`, `stanza`, and `table`. The default value is `table`.

--output-file *OutputFile*
Specifies the name of an output file in which the component data will listed.

--sort *SortKey*
Indicates that the command output is to be sorted by the attribute specified in *SortKey*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlscomp` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. To list components sorted by name, enter the following command:

```
$ mmlscomp --sort name
```

   The system displays output similar to this:

```
    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HEA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      2  1818-80E     SV12345007     G1E1-SX98760044     01
      4  1818-80E     SV12345005     G1E2-SX98760049     05
      3  1818-80E     SV12345003     G1E3-SX98760041     13
      7  1818-80E     SV12345008     G1E4-SX98760036     17
      1  1818-80E     SV12345001     G1E5-SX98760048     21
      6  1818-80E     SV12345002     G1E6-SX98760050     25
      8  1818-80E     SV12345006     G1E7-SX98760039     33
      5  1818-80E     SV12345004     G1E8-SX98760052     37
```

2. To list components in stanza format, enter the following command:

```
mmlscomp --format stanza
```

   The system displays output similar to this:

```
%rack:
  compId=9
  name=R01C01
  partNumber=1410HEA

%storageEnclosure:
  compId=1
  displayId=21
  name='1818-80E-SV12345001'
  partNumber=1818-80E
  serialNumber=SV12345001

%storageEnclosure:
  compId=2
  displayId=1
  name='1818-80E-SV12345007'
  partNumber=1818-80E
  serialNumber=SV12345007

%storageEnclosure:
  compId=3
  displayId=13
  name='G1E3-SX98760044'
  partNumber=1818-80E
  serialNumber=SV12345003
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmlscomploc command

Displays storage component locations.

Synopsis

```
mmlscomploc [--type CompType] [--part-number PartNumber]
            [--serial-number SerialNumber] [--name Name] [--comp-id CompId]
            [--container-id ContainerId] [--format Format]
            [--output-file OutputFile] [--sort SortKey]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmlscomploc` command displays a list of the locations of storage components.

Parameters

--type *CompType*
Specifies a component type, which may include wildcard characters. The resulting list includes those components whose component types (`rack`, `server`, or `storageEnclosure`) match the specified value.

--part-number *PartNumber*
Specifies a part number, which may include wildcard characters. The resulting list includes those components whose part numbers match the specified value.

--serial-number *SerialNumber*
Specifies a serial number, which may include wildcard characters. The resulting list includes those components whose serial numbers match the specified value.

--name *Name*
Specifies a name, which may include wildcard characters. The resulting list includes those components whose names match the specified value.

--comp-id *CompId*
Specifies a component ID, which may include wildcard characters. The resulting list includes those components whose component IDs match the specified value.

--container-id *ContainerId*
Specifies a container ID, which may include wildcard characters. The resulting list includes those components whose container IDs match the specified value.

--format *Format*
Specifies the format in which the component locations will be listed. Valid format values are:

- `bare`
- `colon`
- `csv`
- `none`
- `shell`
- `stanza`
- `table` (default)

`stanza`, and `table`. The default value is `table`.

--output-file *OutputFile*
Specifies the name of an output file in which the component locations will be listed.

--sort *SortKey*
Indicates that the command output is to be sorted by the attribute specified in *SortKey*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlscomploc` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To list component locations, enter the following command:

```
mmlscomploc
```

The system displays output similar to this:

```
Component           Location
------------------  ------------------
1818-80E-SV12345007 Rack R01C01 U01-04
1818-80E-SV12345005 Rack R01C01 U05-08
1818-80E-SV12345003 Rack R01C01 U13-16
1818-80E-SV12345008 Rack R01C01 U17-20
1818-80E-SV12345001 Rack R01C01 U21-24
1818-80E-SV12345002 Rack R01C01 U25-28
```

```
1818-80E-SV12345006 Rack R01C01 U33-36
1818-80E-SV12345004 Rack R01C01 U37-40
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmlscompspec command

Displays storage component specifications.

Synopsis

```
mmlscompspec [--type CompType] [--part-number PartNumber]
             [--format Format] [--output-file OutputFile]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmlscompspec` command displays a list of the specifications of storage components with attributes that match the specified parameters.

Parameters

--type *CompType*
Specifies a component type, which may include wildcard characters. The resulting list includes those components whose component types (`rack`, `server`, or `storageEnclosure`) match the specified value.

--part-number ***PartNumber***
Specifies a part number, which may include wildcard characters. The resulting list includes those components whose part numbers match the specified value.

--format ***Format***
Specifies the format in which the component locations will be listed. Valid values are: `colon`, `column`, `stanza`, and `table`. The default value is `table`.

--output-file ***OutputFile***
Specifies the name of an output file in which the component specifications will be listed.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlscompspec` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To list existing component specifications, enter the following command:

```
mmlscompspec
```

The system displays output similar to this:

```
    Rack Specifications

Part Number  Height  Description
-----------  ------  -------------------------------------------------
1410HEA          42  42U 1200mm Deep Expansion Rack
1410HPA          42  42U 1200mm Deep Primary Rack

    Server Specifications

Part Number  Height  Description
-----------  ------  -------------------
824722L           2 IBM Power System S822L

    Storage Enclosure Specifications

Part Number  Height  Description                 Vendor ID  Product ID  Drive Slots  Has
Display ID
-----------  ------  --------------------------  ---------  ----------  -----------
--------------
1818-80E          4  DCS3700 Expansion Enclosure  IBM        DCS3700
60               1
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddcomp command" on page 264
- "mmaddcompspec command" on page 266
- "mmchcomp command" on page 273
- "mmchcomploc command" on page 275

- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “mmsyncdisplayid command” on page 340
- Chapter 7, “Configuring components on the Elastic Storage Server ,” on page 75.

Location

`/usr/lpp/mmfs/bin`

mmlsenclosure command

Displays the environmental status of IBM Storage Scale RAID disk enclosures.

Synopsis

```
mmlsenclosure {all | SerialNumber}
                  [ -N {all | Node,[Node...] | NodeFile | NodeClass} ]
                  [-L] [--not-ok]
```

Availability

Available with the Elastic Storage Server.

Description

Use the `mmlsenclosure` command to display the environmental status of the IBM Storage Scale RAID disk enclosures in the cluster. The command reports data about enclosure fans, power supplies, and other FRUs so that the equipment can be monitored and serviced in the field. You can choose to display either a brief summary or a more detailed listing of information.

Note: This command displays SCSI Enclosure Services (SES) status only and does not necessarily report on all enclosures.

Output values for mmlsenclosure *SerialNumber* **-L**

Descriptions of the output values for the `mmlsenclosure` *SerialNumber* `-L` command follow, as displayed by row, from top to bottom and left to right.

serial number
Is the serial number of the enclosure.

needs service
Indicates whether any of the components is reporting a failure.

Note: If `enclosure` is the only component that is showing a failure, it is probably because there is an LED light on for least one of the drives. This is *not* an enclosure failure.

nodes
Are the hostnames of the nodes that have access to this enclosure. To see all of the nodes that could have access to the enclosures, you need to specify `-N all` or a list of all of the server nodes.

component type
Is the component type within the enclosure. The component types follow:

currentSensor
Is a current sensor.

dcm
Is a drawer control module.

door
Is a drawer latch.

drawer
Is a drawer.

enclosure
Is a storage enclosure.

esm
Is an environmental service module.

expander
Is a sub-expander.

fan
Is a cooling fan.

powerSupply
Is a power supply.

sideplane
Is an expander board.

tempSensor
Is a temperature sensor.

voltageSensor
Is a voltage sensor.

component id
Is the component ID for the component type.

failed
Indicates whether the component is reporting a failing state.

value
Is the value that is reported by the SES information, where applicable.

unit
Is the measurement unit for the value specified.

properties
Lists additional information, where applicable - about a possible failure, for example.

fru
Is the field replaceable unit associated with the component. This field may be blank for older enclosures.

location
Is the location of the component within the enclosure. This field may be blank for older enclosures.

Note: In the case of failures where the fru is not explicitly listed, the field replaceable units are:

- Drives.
- Enclosure drawers. Any failure that is related to a drawer control module (DCM)'s component ID (component type: `dcm`) requires a drawer replacement.
- Environmental service modules (ESMs).
- Power supplies.
- Fan assemblies.

Parameters

all | *SerialNumber*
Specifies the serial number of the enclosure for which status is to be displayed. `all` specifies that status for all enclosures on the specified nodes is to be displayed.

-N {`all` | *Node*,[*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes from which data is to be gathered. The default is to collect information from the node on which the command is issued.

-L
Specifies that more detailed information is to be displayed. The default for this command is to display summary information only.

--not-ok
Specifies that you wish to display the status of only the components that are reporting an unusual condition.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. To display the status of all enclosures on the current node, run:

```
mmlsenclosure all
```

   The system displays information similar to this:

```
                  needs  nodes
serial number     service
-------------     ------- ------
SV14507574        no      mgmt001st001
SV14519824        no      mgmt001st001
SV15103688        no      mgmt001st001
SV15103765        yes     mgmt001st001
```

2. To display the status of all enclosures on all nodes, run:

```
mmlsenclosure  all -N all
```

   The system displays information similar to this:

```
                  needs  nodes
serial number     service
-------------     ------- ------
SV14507574        no      mgmt001st001,mgmt002st001
SV14519824        no      mgmt001st001,mgmt002st001
SV15103688        no      mgmt001st001,mgmt002st001
SV15103765        yes     mgmt001st001,mgmt002st001
```

3. To display the status of all enclosures on all nodes that are reporting an unusual condition, run:

```
mmlsenclosure all -N all --not-ok
```

The system displays information similar to this:

```
                  needs  nodes
serial number     service
-------------     ------- ------
SV15103765        yes     mgmt001st001,mgmt002st001
```

4. To display a detailed status for the components of enclosure 78E00HL, run:

```
mmlsenclosure 78E00HL -L
The system displays information similar to this:
                   needs  nodes
 serial number     service
 -------------     ------- ------
 78E00HL           no      c202f06fs01a.gpfs.net
 component type  serial number     component id    failed value   unit   properties  fru     location
 --------------  -------------     ------------    ------ -----   ----   ----------  ---     --------
 canister        78E00HL           0               no                    canisterSN=78E00HLb
canister_fru canister1_top
 canister        78E00HL           1               no                    canisterSN=78E00HLa
canister_fru canister2_bottom
 component type  serial number     component id    failed value   unit   properties  fru     location
 --------------  -------------     ------------    ------ -----   ----   ----------  ---     --------
 cpu             78E00HL           0               no                                cpu_fru
canister1_cpu0
 cpu             78E00HL           1               no                                cpu_fru
canister1_cpu1
 cpu             78E00HL           2               no                                cpu_fru
canister2_cpu0
 cpu             78E00HL           3               no                                cpu_fru
canister2_cpu1

 component type  serial number     component id    failed value   unit   properties  fru     location
 --------------  -------------     ------------    ------ -----   ----   ----------  ---     --------
 dimm            78E00HL           0               no                                dimm_fru
canister1_cpu0_channel0a
 dimm            78E00HL           10              no                                dimm_fru
canister1_cpu0_channel5a
 dimm            78E00HL           11              no                    ABSENT      dimm_fru
canister1_cpu0_channel5b
 dimm            78E00HL           12              no                                dimm_fru
canister1_cpu1_channel0a
 dimm            78E00HL           13              no                    ABSENT      dimm_fru
canister1_cpu1_channel0b
 dimm            78E00HL           14              no                                dimm_fru
canister1_cpu1_channel1a
 dimm            78E00HL           15              no                    ABSENT      dimm_fru
canister1_cpu1_channel1b
 dimm            78E00HL           16              no                                dimm_fru
canister1_cpu1_channel2a
 dimm            78E00HL           17              no                    ABSENT      dimm_fru
canister1_cpu1_channel2b
 dimm            78E00HL           18              no                                dimm_fru
canister1_cpu1_channel3a
 dimm            78E00HL           19              no                    ABSENT      dimm_fru
canister1_cpu1_channel3b
 dimm            78E00HL           1               no                    ABSENT      dimm_fru
canister1_cpu0_channel0b
 dimm            78E00HL           20              no                                dimm_fru
canister1_cpu1_channel4a
 dimm            78E00HL           21              no                    ABSENT      dimm_fru
canister1_cpu1_channel4b
 dimm            78E00HL           22              no                                dimm_fru
canister1_cpu1_channel5a
 dimm            78E00HL           23              no                    ABSENT      dimm_fru
canister1_cpu1_channel5b
 dimm            78E00HL           24              no                                dimm_fru
canister2_cpu0_channel0a
 dimm            78E00HL           25              no                    ABSENT      dimm_fru
canister2_cpu0_channel0b
 dimm            78E00HL           26              no                                dimm_fru
```

```
canister2_cpu0_channel1a
 dimm            78E00HL          27              no                    ABSENT      dimm_fru
canister2_cpu0_channel1b
 dimm            78E00HL          28              no                                dimm_fru
canister2_cpu0_channel2a
 dimm            78E00HL          29              no                    ABSENT      dimm_fru
canister2_cpu0_channel2b
 dimm            78E00HL          2               no                                dimm_fru
canister1_cpu0_channel1a
 dimm            78E00HL          30              no                                dimm_fru
canister2_cpu0_channel3a
 dimm            78E00HL          31              no                    ABSENT      dimm_fru
canister2_cpu0_channel3b
 dimm            78E00HL          32              no                                dimm_fru
canister2_cpu0_channel4a
 dimm            78E00HL          33              no                    ABSENT      dimm_fru
canister2_cpu0_channel4b
 dimm            78E00HL          34              no                                dimm_fru
canister2_cpu0_channel5a
 dimm            78E00HL          35              no                    ABSENT      dimm_fru
canister2_cpu0_channel5b
 dimm            78E00HL          36              no                                dimm_fru
canister2_cpu1_channel0a
 dimm            78E00HL          37              no                    ABSENT      dimm_fru
canister2_cpu1_channel0b
 dimm            78E00HL          38              no                                dimm_fru
canister2_cpu1_channel1a
 dimm            78E00HL          39              no                    ABSENT      dimm_fru
canister2_cpu1_channel1b
 dimm            78E00HL          3               no                    ABSENT      dimm_fru
canister1_cpu0_channel1b
 dimm            78E00HL          40              no                                dimm_fru
canister2_cpu1_channel2a
 dimm            78E00HL          41              no                    ABSENT      dimm_fru
canister2_cpu1_channel2b
 dimm            78E00HL          42              no                                dimm_fru
canister2_cpu1_channel3a
 dimm            78E00HL          43              no                    ABSENT      dimm_fru
canister2_cpu1_channel3b
 dimm            78E00HL          44              no                                dimm_fru
canister2_cpu1_channel4a
 dimm            78E00HL          45              no                    ABSENT      dimm_fru
canister2_cpu1_channel4b
 dimm            78E00HL          46              no                                dimm_fru
canister2_cpu1_channel5a
 dimm            78E00HL          47              no                    ABSENT      dimm_fru
canister2_cpu1_channel5b
 dimm            78E00HL          4               no                                dimm_fru
canister1_cpu0_channel2a
 dimm            78E00HL          5               no                    ABSENT      dimm_fru
canister1_cpu0_channel2b
 dimm            78E00HL          6               no                                dimm_fru
canister1_cpu0_channel3a
 dimm            78E00HL          7               no                    ABSENT      dimm_fru
canister1_cpu0_channel3b
 dimm            78E00HL          8               no                                dimm_fru
canister1_cpu0_channel4a
 dimm            78E00HL          9               no                    ABSENT      dimm_fru
canister1_cpu0_channel4b

 component type  serial number    component id    failed value   unit   properties  fru     location
 --------------  -------------    ------------    ------ -----   ----   ----------  ---     --------
 enclosure       78E00HL          ONLY            no                                midplane_fru
middle_chassis
 component type  serial number    component id    failed value   unit   properties  fru     location
 --------------  -------------    ------------    ------ -----   ----   ----------  ---     --------
 expander        78E00HL          0               no                                midplane_fru
middle_chassis
 component type  serial number    component id    failed value   unit   properties  fru     location
 --------------  -------------    ------------    ------ -----   ----   ----------  ---     --------
 fan             78E00HL          0               no     20460   RPM                fan_fru
canister1_fan1_module1
 fan             78E00HL          10              no     20460   RPM                fan_fru
canister2_fan1_module1
 fan             78E00HL          11              no     17550   RPM                fan_fru
canister2_fan2_module1
 fan             78E00HL          12              no     20460   RPM                fan_fru
canister2_fan1_module2
 fan             78E00HL          13              no     17550   RPM                fan_fru
canister2_fan2_module2
 fan             78E00HL          14              no     20460   RPM                fan_fru
canister2_fan1_module3
```

```
 fan             78E00HL           15              no     17550   RPM               fan_fru
canister2_fan2_module3
 fan             78E00HL           16              no     20460   RPM               fan_fru
canister2_fan1_module4
 fan             78E00HL           17              no     17550   RPM               fan_fru
canister2_fan2_module4
 fan             78E00HL           18              no     20460   RPM               fan_fru
canister2_fan1_module5
 fan             78E00HL           19              no     17550   RPM               fan_fru
canister2_fan2_module5
 fan             78E00HL           1               no     17550   RPM               fan_fru
canister1_fan2_module1
 fan             78E00HL           20              no     2000    RPM               fan_fru psu1_fan1
 fan             78E00HL           21              no     2030    RPM               fan_fru psu1_fan2
 fan             78E00HL           22              no     1950    RPM               fan_fru psu2_fan1
 fan             78E00HL           23              no     2030    RPM               fan_fru psu2_fan2
 fan             78E00HL           2               no     20460   RPM               fan_fru
canister1_fan1_module2
 fan             78E00HL           3               no     17550   RPM               fan_fru
canister1_fan2_module2
 fan             78E00HL           4               no     20460   RPM               fan_fru
canister1_fan1_module3
 fan             78E00HL           5               no     17550   RPM               fan_fru
canister1_fan2_module3
 fan             78E00HL           6               no     20460   RPM               fan_fru
canister1_fan1_module4
 fan             78E00HL           7               no     17550   RPM               fan_fru
canister1_fan2_module4
 fan             78E00HL           8               no     20460   RPM               fan_fru
canister1_fan1_module5
 fan             78E00HL           9               no     17550   RPM               fan_fru
canister1_fan2_module5
 component type  serial number     component id    failed value   unit   properties  fru     location
 --------------  -------------     ------------    ------ -----   ----   ----------  ---     --------
 powerSupply     78E00HL           0               no                                psu_fru psu1_left
 powerSupply     78E00HL           1               no                                psu_fru psu2_right

 component type  serial number     component id    failed value   unit   properties  fru     location
 --------------  -------------     ------------    ------ -----   ----   ----------  ---     --------
 tempSensor      78E00HL           0               no     31      C                  midplane_fru
ambient_air1
 tempSensor      78E00HL           10              no     41      C                  canister_fru
canister1_pci_slot1
 tempSensor      78E00HL           11              no     39      C                  canister_fru
canister1_pci_slot2
 tempSensor      78E00HL           12              no     36      C                  canister_fru
canister1_pci_slot3
 tempSensor      78E00HL           13              no     43      C                  canister_fru
canister1_pcix
 tempSensor      78E00HL           14              no     41      C                  canister_fru
canister1_cpu0
 tempSensor      78E00HL           15              no     38      C                  canister_fru
canister1_cpu1
 tempSensor      78E00HL           16              no     34      C                  canister_fru
canister1_cpu0_channel0a
 tempSensor      78E00HL           17              no             C      ABSENT      canister_fru
canister1_cpu0_channel0b
 tempSensor      78E00HL           18              no     34      C                  canister_fru
canister1_cpu0_channel1a
 tempSensor      78E00HL           19              no             C      ABSENT      canister_fru
canister1_cpu0_channel1b
 tempSensor      78E00HL           1               no     31      C                  midplane_fru
ambient_air2
 tempSensor      78E00HL           20              no     34      C                  canister_fru
canister1_cpu0_channel2a
 tempSensor      78E00HL           21              no             C      ABSENT      canister_fru
canister1_cpu0_channel2b
 tempSensor      78E00HL           22              no     38      C                  canister_fru
canister1_cpu0_channel3a
 tempSensor      78E00HL           23              no             C      ABSENT      canister_fru
canister1_cpu0_channel3b
 tempSensor      78E00HL           24              no
```

See also

See also the following *IBM Storage Scale RAID: Administration* topic:

- “mmchenclosure command” on page 277.

Location

/usr/lpp/mmfs/bin

mmlsfirmware command

Displays the current firmware level of storage components.

Synopsis

```
mmlsfirmware [ --type {storage-enclosure | drive | host-adapter} ]
             [ --serial-number SerialNumber ] [--not-latest]
             [ -N {Node[,Node...] | NodeFile | NodeClass} ]
```

Availability

Available with the Elastic Storage Server.

Description

Use the `mmlsfirmware` command to display the current firmware levels of storage components. By default, the `mmlsfirmware` command collects information from the node on which it is issued and displays the firmware levels for all component types.

An asterisk (*) prepended to the available firmware value indicates that newer firmware is available. In some cases, the available firmware level might have an asterisk even though it matches the current firmware level. This indicates that a subcomponent requires updating.

Parameters

--type { storage-enclosure | drive | host-adapter }
Displays the firmware levels for a specific component type.

--serial-number *SerialNumber*
Displays the firmware levels for the storage enclosure with the specified serial number.

--not-latest
Displays the components for which there are available updates or for which there are no available updates.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes from which to gather firmware data.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

1. To display the firmware level of all drives, storage enclosures, and host adapters on the current node, issue this command:

```
mmlsfirmware
```

The system displays information similar to this:

```
                        enclosure       firmware   available
type       product id   serial number   level      firmware    location
----       ----------   -------------   --------   --------    --------
enclosure  DCS3700      0123456789AB    039A,039A  039A        Rack BB1RACK U01-04
enclosure  DCS3700      SV11812206      039A,039A  039A        Rack BB1RACK U13-16
enclosure  DCS3700      SV12616296      039A,039A  039A        Rack BB1RACK U05-08
enclosure  DCS3700      SV13306129      039A,039A  039A        Rack BB1RACK U17-20

                    enclosure      firmware  available
type   product id   serial number  level     firmware  location
----   ----------   -------------  --------  --------  --------
drive  ST2000NM0001 0123456789AB   BC4B      BC4B      Rack BB1RACK U01-04, Enclosure 1818-80E-0123456789AB
                                                       Drawer 1 Slot 10
drive  ST2000NM0001 0123456789AB   BC4B      BC4B      Rack BB1RACK U01-04, Enclosure 1818-80E-0123456789AB
                                                       Drawer 1 Slot 11
drive  ST2000NM0001 0123456789AB   BC4B      BC4B      Rack BB1RACK U01-04, Enclosure 1818-80E-0123456789AB
                                                       Drawer 1 Slot 12
.
.
.

         product   firmware      available     bios          available     UEFI         available
type     id        level         firmware      level         bios          level        UEFI          location
----     -------   --------      ---------     --------      ---------     --------     ---------     --------
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.27.01.01  07.27.01.01   c55f04n03 0 00:52:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.27.01.01  07.27.01.01   c55f04n03 1 00:54:00:00
adapter  0x3070    20.00.00.00   *20.00.04.00  07.35.00.00   *07.39.00.00  07.21.01.00  *07.27.01.01  c55f04n03 2 00:03:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.21.01.00  *07.27.01.01  c55f04n03 3 00:05:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.22.04.03  *07.27.01.01  c55f04n03 4 00:03:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.22.04.03  *07.27.01.01  c55f04n03 5 00:05:00:00
```

Note: For adapters, the asterisk (*), which is prepended to the `available firmware`, `available bios`, and `available UEFI` values, indicates that newer firmware is available.

2. To display the components on the current node that are not at the latest available firmware levels, issue this command:

```
mmlsfirmware --not-latest
```

The system displays information similar to this:

```
       product       enclosure     firmware available
type   id            serial number level    firmware      location
----   ----------    ------------- -------- ---------     --------
drive  SDLKOEDM-200GL 0123456789AB  HD33     not_available Rack BB1RACK U01-04, Enclosure 1818-80E-0123456789AB Drawer 1 Slot 3
drive  SDLKOEDM-200GL 0123456789AB  HD33     not_available Rack BB1RACK U01-04, Enclosure 1818-80E-0123456789AB Drawer 5 Slot 12
.
.
.

         product   firmware      available     bios          available     UEFI         available
type     id        level         firmware      level         bios          level        UEFI          location
----     -------   --------      ---------     --------      ---------     --------     ---------     --------
adapter  0x3070    20.00.00.00   *20.00.04.00  07.35.00.00   *07.39.00.00  07.21.01.00  *07.27.01.01  c55f04n03 2 00:03:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.21.01.00  *07.27.01.01  c55f04n03 3 00:05:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.22.04.03  *07.27.01.01  c55f04n03 4 00:03:00:00
adapter  0x3070    20.00.04.00   20.00.04.00   07.39.00.00   07.39.00.00   07.22.04.03  *07.27.01.01  c55f04n03 5 00:05:00:00
```

Note:

a. For adapters, the asterisk (*), which is prepended to the `available firmware`, `available bios`, and `available UEFI` values, indicates that newer firmware is available.

b. Firmware was not available for the `mmchfirmware` command to load for the two drives in this example. This would be an unexpected situation.

3. To display the firmware level of the storage enclosures on the current node, issue this command:

```
mmlsfirmware --type storage-enclosure
```

The system displays information similar to this:

```
                        enclosure          firmware     available
type       product id   serial number      level        firmware    location
----       ----------   -------------      --------      --------   --------
enclosure  DCS3700      SV11933001         039A,039A     039A       Rack
```

```
BB1RACK U01-04
enclosure  DCS3700         SV11812206         039A,039A     039A       Rack
BB1RACK U13-16
enclosure  DCS3700         SV12616296         039A,039A     039A       Rack
BB1RACK U05-08
enclosure  DCS3700         SV13306129         039A,039A     039A       Rack
BB1RACK U17-20
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmsyncdisplayid command” on page 340.

Location

`/usr/lpp/mmfs/bin`

mmlsnvmestatus

The **mmlsnvmestatus** command is used to list the status of all the NVMe controllers or a specific NVMe controller.

Synopsis

```
mmlsnvmestatus {all | SerialNumber}
[-N {Node,[Node...] | NodeFile | NodeClass}]
[--not-ok][-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the command to list the status of all the NVMe controllers or a specific NVMe controller of all the nodes that are specified, or the nodes of the node file or node class specified. By default, if no nodes are specified, the **mmlsnvmestatus** command finds the server nodes of all the configured Recovery Groups (RGs) and list the status of the NVMe controller of those nodes.

Use the **mmlsnvmestatus** command to perform the following functions on the specified node:

- To find the status of all NVMe controller or a specific NVMe controller.
- To display the status of only the NVMe controllers that are reporting an unusual condition.

- To view the individual categorize and summary status of NVMe controller in colon separated form.

Parameters

all | *SerialNumber*
Specifies that the status of NVMe controller must be displayed. The option `all` specifies that the status of all NVMe controllers on the specified nodes must be displayed. *SerialNumber* indicates the serial number of the NVMe controller for which the status must be displayed.

-N {*Node*, [*Node*...]|*NodeFile* | *NodeClass*}
Specifies the list of nodes or nodes of node file or node class from which NVMe controller status must be displayed. By default, the NVMe controller status information from all the server nodes of the all Recovery Groups (RGs) configured is displayed.

--not-ok
Specifies to display the status of only those NVMe controllers that are reporting an unusual condition.

-Y

Specifies that the output needs to be displayed in colon separated form. The output displays the colon separated status of all the individual categories, including the summary status of the NVMe controllers.

If -Y is specified, all individual NVMe controller's status including the summary status is displayed as shown in the following example.

```
[root@c145f02n01 bin]# mmlsnvmestatus all -Y
mmlsnvmestatus:vaultBackup:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
spareSpace:availableSpare:usedSpare:
mmlsnvmestatus:tempSensor:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
operationalTemp:cpuTemp:systemTemp:supercapTemp:
mmlsnvmestatus:backupFail:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
vaultOperationFail:operationalMode:mediaErrors:
mmlsnvmestatus:ioMode:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
readOnly:backupPowerLow:backupInProgress:noSpareBlocks:
mmlsnvmestatus:ioOperations:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
dataRead:dataWritten:cmdsRead:cmdsWritten:
mmlsnvmestatus:deviceOpMode:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
operationalMode:mediaErrors:
mmlsnvmestatus:summary:HEADER:version:reserved:reserved:node:NVMeCtrl:serialNumber:
status:needsService:permReadOnly:tempReadOnly:OptimalLinkState:OptimalLBAFormats
mmlsnvmestatus:backupFail:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NO:NORMAL:0:
mmlsnvmestatus:backupFail:0:1:::c145f02n02:/dev/nvme0:04000017F900:NO:NORMAL:0:
mmlsnvmestatus:deviceOpMode:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NORMAL:0:
mmlsnvmestatus:deviceOpMode:0:1:::c145f02n02:/dev/nvme0:04000017F900:NORMAL:0:
mmlsnvmestatus:ioMode:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NO:NO:NO:NO:
mmlsnvmestatus:ioMode:0:1:::c145f02n02:/dev/nvme0:04000017F900:NO:NO:NO:NO:
mmlsnvmestatus:ioOperations:0:1:::c145f02n01:/dev/
nvme0:040000185FD2:637330:89904863:4263413:3680839965:
mmlsnvmestatus:ioOperations:0:1:::c145f02n02:/dev/
nvme0:04000017F900:634037:76631284:4164896:3499213369:
mmlsnvmestatus:summary:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NORMAL:NO:NO:NO:YES:YES:
mmlsnvmestatus:summary:0:1:::c145f02n02:/dev/nvme0:04000017F900:NORMAL:NO:NO:NO:YES:YES:
mmlsnvmestatus:tempSensor:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NORMAL:313:306:298:
mmlsnvmestatus:tempSensor:0:1:::c145f02n02:/dev/nvme0:04000017F900:NORMAL:313:307:301:
mmlsnvmestatus:vaultBackup:0:1:::c145f02n01:/dev/nvme0:040000185FD2:NORMAL:98:2:
mmlsnvmestatus:vaultBackup:0:1:::c145f02n02:/dev/nvme0:04000017F900:NORMAL:99:1:
```

If -Y option is not specified, then the **mmlsnvmestatus** command displays human readable summary status of the NVMe controllers as shown in the following example.

```
[root@c145f02n01 bin]# mmlsnvmestatus all
                                          Optimal     Optimal      needs
 node        NVMe device    serial number  Link State  LBS Formats  service
 ------      -----------    -------------  ----------  -----------  -------
 c145f02n01  /dev/nvme0     040000185FD2   YES         YES          NO
 c145f02n02  /dev/nvme0     04000017F900   YES         YES          NO
[root@c145f02n01 bin]#
```

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmlsnvmestatus** command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see "Requirements for administering IBM Storage Scale RAID" on page 17.

Example

The following example shows the usage of **mmlsnvmestatus** command to view status of a NVMe controller whose serial number is specified.

```
[root@c202f06fs03a fab_tools]# mmlsnvmestatus  S43RNE0KC00152
                                                Optimal     Optimal      needs
 node          NVMe device    serial number     Link State  LBA Format   service
 ------        -----------    -------------     ----------  ----------   -------
 c202f06fs03a  /dev/nvme8     S43RNE0KC00152    YES         YES          NO
[root@c202f06fs03a fab_tools]#
```

See also

See also the following IBM Storage Scale RAID: Administration topics.

- "mmlspdisk command" on page 324
- "mmlsrecoverygroup command" on page 327

Location

`/usr/lpp/mmfs/bin`

mmlspdisk command

Lists information for one or more IBM Storage Scale RAID pdisks.

Synopsis

```
mmlspdisk {all | RecoveryGroupName [--declustered-array DeclusteredArrayName | --pdisk
pdiskName]}
          [--not-in-use | --not-ok | --replace | --ssd-endurance-percentage
PercentageUsedThreshold]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmlspdisk` command lists information for one or more pdisks, which can be specified in various ways.

Parameters

all | *RecoveryGroupName*
Specifies the recovery group for which the pdisk information is to be listed.

`all` specifies that pdisk information for all recovery groups is to be listed.

RecoveryGroupName specifies the name of a particular recovery group for which the pdisk information is to be listed.

--declustered-array *DeclusteredArrayName*
Specifies the name of a declustered array for which the pdisk information is to be listed.

--pdisk *pdiskName*
Specifies the name of a single pdisk for which the information is to be listed.

--not-in-use
Indicates that information is to be listed only for pdisks that are draining.

--not-ok
Indicates that information is to be listed only for pdisks that are not functioning correctly.

--replace
Indicates that information is to be listed only for pdisks in declustered arrays that are marked for replacement.

--ssd-endurance-percentage *PercentageUsedThreshold*
Shows SSDs that have reached or exceeded the specified write endurance *PercentageUsedThreshold*. The *PercentageUsedThreshold* must be a number between 0 and 100.

Note: SSDs have a finite lifetime based on the number of drive writes per day. The **`ssd-endurance-percentage`** values actually reported will be a number between 0 and 255. This value indicates the percentage of life that is used by the drive. The value 0 indicates that full life remains, and 100 indicates that the drive is at or past its end of life. The drive must be replaced when the value exceeds 100.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlspdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

1. The following command example shows how to display the details regarding pdisk c112d3 in recovery group 000DE37BOT:

```
mmlspdisk 000DE37BOT --pdisk c112d3
```

The system displays output similar to the following:

```
pdisk:
  replacementPriority = 1000
  name = "e2s23"
  device = "/dev/sdbs,/dev/sdcq"
```

```
  recoveryGroup = "rg0"
  declusteredArray = "p30da_d"
  state = "ok"
  capacity  = 299842404352
  freeSpace = 299573968896
  fru = "49Y1840"
  location = "SX31700361-23"
  WWN = "naa.5000C50067A91EC3"
  server = "perseus30ib.almaden.ibm.com"
  reads = 8
  writes = 5
  bytesReadInGiB = 0.002
  bytesWrittenInGiB = 0.001
  IOErrors = 0
  IOTimeouts = 0
  mediaErrors = 0
  checksumErrors = 0
  pathErrors = 0
  relativePerformance = 1.000
  bitErrorRate = 0.000e+00
  rgIndex = 46
  userLocation = ""
  userCondition = "normal"
  hardware = "IBM-ESXS ST9300605SS B559 6XP5GQMP0000M338BUSD"
  hardwareType = Rotating 10000
  nPaths = 2 active (2 expected) 4 total (4 expected)
  nsdFormatVersion = NSD version 2
  paxosAreaOffset = 600122941440
  paxosAreaSize = 4194304
  logicalBlockSize = 4096
  sedSupported = Yes
```

2. To show which pdisks in recovery group 000DE37B0T need replacing:

```
mmlspdisk 000DE37BOT --replace
```

The system displays output similar to the following:

```
pdisk:
   replacementPriority = 0.98
   name = "c052d1"
   device = "/dev/rhdisk556,/dev/rhdisk460"
   recoveryGroup = "000DE37BOT"
   declusteredArray = "DA1"
   state = "dead/systemDrain/noRGD/noVCD/replace"
   .
   .
   .
pdisk:
   replacementPriority = 0.98
   name = "c096d1"
   device = "/dev/rhdisk508,/dev/rhdisk412"
   recoveryGroup = "000DE37BOT"
   declusteredArray = "DA1"
   state = "dead/systemDrain/noRGD/noVCD/replace"
   .
   .
   .
```

3. The following example shows how to display the SSDs in recovery group rgL that have a write endurance percentage that is greater than 50%. Run the following command:

```
mmlspdisk rgL --ssd-endurance-percentage 50
```

The system displays output similar to the following:

```
pdisk:
      replacementPriority = 1000
      name = "e2s078"
      device = "//c145f01n03/dev/sdao,//c145f01n03/dev/sdis,//c145f01n04/dev/
sdbk(notEnabled)"
      recoveryGroup = "rgL"
      declusteredArray = "DA1"
      .
      .
```

```
.
nsdFormatVersion = 2
paxosAreaOffset = 9931033804800
paxosAreaSize = 4194304
logicalBlockSize = 4096
ssdEndurancePercentage = 55       # values range from 0 to 255
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmchpdisk command" on page 283
- "mmdelpdisk command" on page 299
- "mmlsrecoverygroup command" on page 327
- "mmlsrecoverygroupevents command" on page 335
- "mmlsvdisk command" on page 337

Location

`/usr/lpp/mmfs/bin`

mmlsrecoverygroup command

Lists information about IBM Storage Scale RAID recovery groups.

Synopsis

```
mmlsrecoverygroup [ RecoveryGroupName [-L [--pdisk] ] ]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmlsrecoverygroup` command lists information about recovery groups. The command displays various levels of information, depending on the parameters specified.

1. Output values for mmlsrecoverygroup

Descriptions of the output values for the **mmlsrecoverygroup** command follow.

recovery group
Is the name of the recovery group.

declustered arrays with vdisks
Is the number of declustered arrays with vdisks in this recovery group.

vdisks
Is the number of vdisks in this recovery group.

servers
Is the server pair for the recovery group. The intended primary server is listed first, followed by the intended backup server.

2. Output values for mmlsrecoverygroup *RecoveryGroupName*

Descriptions of the output values for the **mmlsrecoverygroup** *RecoveryGroupName* command follow, as displayed by row, from top to bottom and left to right.

recovery group
Is the name of the recovery group.

declustered arrays with vdisks
Is the number of declustered arrays with vdisks in this recovery group.

vdisks
Is the number of vdisks in this recovery group.

servers
Is the server pair for the recovery group. The intended primary server is listed first, followed by the intended backup server.

declustered array with vdisks
Is the name of the declustered array.

vdisks
Is the number of vdisks in this declustered array.

vdisk
Is the name of the vdisk.

RAID code
Is the RAID code for this vdisk.

declustered array
Is the declustered array for this vdisk.

remarks
Indicates the special vdisk type. Only those vdisks with a dedicated function within the recovery group are indicated here: the log, log tip, log tip backup, and log reserved vdisks. This field is blank for file system NSD vdisks.

3. Output values for mmlsrecoverygroup *RecoveryGroupName* **-L**

Descriptions of the output values for the **mmlsrecoverygroup** *RecoveryGroupName* -L command follow, as displayed by row, from top to bottom and left to right.

Recovery group section:

recovery group
Is the name of the recovery group.

declustered arrays
Is the number of declustered arrays in this recovery group.

vdisks
Is the number of vdisks in this recovery group.

pdisks
Is the number of pdisks in this recovery group.

format version
Is the recovery group version.

Declustered array section:

declustered array
Is the name of the declustered array.

needs service
Indicates whether this declustered array needs service. A yes value means that disks need to be replaced.

vdisks
Is the number of vdisks in this declustered array.

pdisks
Is the number of pdisks in this declustered array.

spares
The first number of the pair is the amount of spare space that is reserved for rebuilding, expressed as an equivalent number of pdisks. This spare space is allocated equally among all of the pdisks in the declustered array.

The second number of the pair is the number of vdisk configuration data (VCD) replicas that are maintained across all of the pdisks of the declustered array. This is the internal IBM Storage Scale RAID metadata for the recovery group. The number of these VCD spares is set during recovery group creation and should not be changed.

replace threshold
Is the number of pdisks that must fail before the declustered array reports that it needs service and the pdisks are marked for required replacement.

bit error rate (BER)
If the bit error rate of a disk within a DA is predicted to be beyond an expected threshold, that pdisk will be marked as failing.

enable
Means that the bit error rate enforcement is active for this declustered array. This is the default setting for declustered arrays that support bit error rate enforcement.

disable
Means that the bit error rate enforcement is not active for this declustered array. This setting indicates that bit error rate enforcement has been temporarily disabled.

N/A
Means that bit error rate enforcement is not supported on this declustered array. Bit error rate enforcement is permanently disabled.

free space
Is the amount of raw space in this declustered array that is unused and is available for creating vdisks. The pdisk spare space for rebuild has already been removed from this number. The size of the vdisks that can be created using the raw free space depends on the redundancy requirements of the vdisk RAID code: a 4WayReplicated vdisk of size N uses 4N raw space; an 8+3P vdisk of size N uses 1.375N raw space.

scrub duration
Is the length of time in days over which the scrubbing of all vdisks in the declustered array will complete.

background activity

task
Is the task that is being performed on the declustered array.

inactive
Means that there are no vdisks defined or the declustered array is not currently available.

scrub
Means that vdisks are undergoing routine data integrity maintenance.

rebuild-critical
Means that vdisk tracks with no remaining redundancy are being rebuilt.

rebuild-1r
Means that vdisk tracks with one remaining redundancy are being rebuilt.

rebuild-2r
Means that vdisk tracks with two remaining redundancies are being rebuilt.

progress
Is the completion percentage of the current task.

priority
Is the priority given the current task. Critical rebuilds are given high priority; all other tasks have low priority.

Vdisk section:

vdisk
Is the name of the vdisk.

RAID code
Is the RAID code for this vdisk.

declustered array
Is the declustered array in which this vdisk is defined.

vdisk size
Is the usable size this vdisk.

block size
Is the block (track) size for this vdisk.

checksum granularity
Is the buffer granularity at which IBM Storage Scale RAID checksums are maintained for this vdisk. This value is set automatically at vdisk creation. For file system NSD vdisks, this value is 32 KiB. For log, log tip, log tip backup, and log reserved vdisks on Power Systems servers, this value is 4096.

state
Is the maintenance task that is being performed on this vdisk.

ok
Means that the vdisk is being scrubbed or is waiting to be scrubbed. Only one vdisk in a DA is scrubbed at a time.

1/3-deg
Means that tracks with one fault from three redundancies are being rebuilt.

2/3-deg
Means that tracks with two faults from three redundancies are being rebuilt.

1/2-deg
Means that tracks with one fault from two redundancies are being rebuilt.

critical
Means that tracks with no remaining redundancy are being rebuilt.

inactive
Means that the declustered array that is associated with this vdisk is inactive.

remarks
Indicates the special vdisk type. Only those vdisks with a dedicated function within the recovery group are indicated here: the log, log tip, log tip backup, and log reserved vdisks. This field is blank for file system NSD vdisks.

Fault tolerance section:

config data
Is the type of internal IBM Storage Scale RAID recovery group metadata for which fault tolerance is being reported.

rebuild space
Indicates the space that is available for IBM Storage Scale RAID metadata relocation.

declustered array
Is the name of the declustered array.

VCD spares
Is the number of VCD spares that are defined for the declustered array.

actual rebuild spare space
Is the number of pdisks that are currently eligible to hold VCD spares.

remarks
Indicates the effect or limit the VCD spares have on fault tolerance.

config data
Is the type of internal IBM Storage Scale RAID recovery group metadata for which fault tolerance is being reported.

rg descriptor
Indicates the recovery group declustered array and pdisk definitions.

system index
Indicates the vdisk RAID stripe partition definitions.

max disk group fault tolerance
Shows the theoretical maximum fault tolerance for the `config data`.

actual disk group fault tolerance
Shows the current actual fault tolerance for the `config data`, given the current state of the recovery group, its pdisks, and its number, type, and size of vdisks.

remarks
Indicates whether the actual fault tolerance is limiting other fault tolerances or is limited by other fault tolerances.

vdisk
Is the name of the vdisk for which fault tolerance is being reported.

max disk group fault tolerance
Shows the theoretical maximum fault tolerance for the vdisk.

actual disk group fault tolerance
Shows the current actual fault tolerance for the vdisk, given the current state of the recovery group, its pdisks, and its number, type, and size of vdisks.

remarks
Indicates why the actual fault tolerance might be different from the maximum fault tolerance.

Server section:

active recovery group server
Is the currently-active recovery group server.

servers
Is the server pair for the recovery group. The intended primary server is listed first, followed by the intended backup server.

4. Output values for mmlsrecoverygroup *RecoveryGroupName* **-L --pdisk**

Descriptions of the output values for the **mmlsrecoverygroup** *RecoveryGroupName* `-L --pdisk` command follow, as displayed by row, from top to bottom and left to right.

Recovery group section:

recovery group
Is the name of the recovery group.

declustered arrays
Is the number of declustered arrays in this recovery group.

vdisks
Is the number of vdisks in this recovery group.

pdisks
Is the number of pdisks in this recovery group.

format version
Is the recovery group version.

Declustered array section:

declustered array
Is the name of the declustered array.

needs service
Indicates whether this declustered array needs service. A `yes` value means that disks need to be replaced.

vdisks
Is the number of vdisks in this declustered array.

pdisks
Is the number of pdisks in this declustered array.

spares
The first number of the pair is the amount of spare space that is reserved for rebuilding, expressed as an equivalent number of pdisks. This spare space is allocated equally among all of the pdisks in the declustered array.

The second number of the pair is the number of VCD replicas that are maintained across all of the pdisks of the declustered array. This is the internal IBM Storage Scale RAID metadata for the recovery group. The number of these VCD spares is set during recovery group creation and should not be changed.

replace threshold
Is the number of pdisks that must fail before the declustered array reports that it needs service and the pdisks are marked for required replacement.

free space
Is the amount of raw space in this declustered array that is unused and is available for creating vdisks. The pdisk spare space for rebuilding has already been removed from this number. The size of the vdisks that can be created using the raw free space depends on the redundancy requirements of the vdisk RAID code: a 4WayReplicated vdisk of size N uses 4N raw space; an 8+3P vdisk of size N uses 1.375N raw space.

scrub duration
Is the length of time in days over which the scrubbing of all vdisks in the declustered array will complete.

background activity

task
Is the task that is being performed on the declustered array.

inactive
Means that there are no vdisks defined or the declustered array is not currently available.

scrub
Means that vdisks are undergoing routine data integrity maintenance.

rebuild-critical
Means that vdisk tracks with no remaining redundancy are being rebuilt.

rebuild-1r
Means that vdisk tracks with only one remaining redundancy are being rebuilt.

rebuild-2r
Means that vdisk tracks with two remaining redundancies are being rebuilt.

progress
Is the completion percentage of the current task.

priority
Is the priority given the current task. Critical rebuilds are given high priority; all other tasks have low priority.

Pdisk section:

pdisk
Is the name of the pdisk.

n. active, total paths
Indicates the number of active (in-use) and total block device paths to the pdisk. The total paths include the paths that are available on the standby server.

declustered array
Is the name of the declustered array to which the pdisk belongs.

free space
Is the amount of raw free space that is available on the pdisk.

Note: A pdisk that has been taken out of service and completely drained of data will show its entire capacity as free space, even though that capacity is not available for use.

user condition
Is the condition of the pdisk from the system administrator's persepective. A "normal" condition requires no attention, while "replaceable" means that the pdisk might be, but is not necessarily required to be, physically replaced.

state, remarks
Is the state of the pdisk. For a description of pdisk states, see the topic "Pdisk states" on page 57 in *IBM Storage Scale RAID: Administration*.

Server section:

active recovery group server
Is the currently-active recovery group server.

servers
Is the server pair for the recovery group. The intended primary server is listed first, followed by the intended backup server.

Parameters

RecoveryGroupName
Specifies the recovery group for which the information is being requested. If no other parameters are specified, the command displays only the information that can be found in the GPFS cluster configuration data.

-L
Displays more detailed runtime information for the specified recovery group.

--pdisk
Indicates that pdisk information is to be listed for the specified recovery group.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlsrecoverygroup` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. The following command example shows how to list all the recovery groups in the GPFS cluster:

```
mmlsrecoverygroup
```

The system displays output similar to the following:

```
                    declustered
                    arrays with
recovery group        vdisks     vdisks  servers
------------------  -----------  ------  -------
BB1RGL                        4       8  c45f01n01-ib0.gpfs.net,c45f01n02-ib0.gpfs.net
BB1RGR                        3       7  c45f01n02-ib0.gpfs.net,c45f01n01-ib0.gpfs.net
```

2. The following command example shows how to list the basic non-runtime information for recovery group 000DE37B0T:

```
mmlsrecoverygroup 000DE37B0T
```

The system displays output similar to the following:

```
                    declustered
                    arrays with
recovery group        vdisks     vdisks  servers
------------------  -----------  ------  -------
BB1RGL                      4       8  c45f01n01-ib0.gpfs.net,c45f01n02-ib0.gpfs.net

declustered array
    with vdisks     vdisks
------------------  ------
DA1                      3
DA2                      3
NVR                      1
SSD                      1

                                          declustered
vdisk               RAID code               array     remarks
------------------  ------------------  -----------  -------
BB1RGLDATA1         8+3p                DA1
BB1RGLDATA2         8+3p                DA2
BB1RGLMETA1         4WayReplication     DA1
BB1RGLMETA2         4WayReplication     DA2
lhome_BB1RGL        4WayReplication     DA1          log
ltbackup_BB1RGL     Unreplicated        SSD
ltip_BB1RGL         2WayReplication     NVR
reserved1_BB1RGL    4WayReplication     DA2
```

3. The following command example shows how to display the runtime status of recovery group BB01L:

```
mmlsrecoverygroup BB01L -L
```

The system displays output similar to the following:

```
                 declustered                   current       allowable
recovery group      arrays     vdisks  pdisks  format version format version
-----------------  -----------  ------  ------ -------------- --------------
BB01L                     1       5      49  5.1.2.0        5.1.2.0

declustered   needs                          replace                        scrub       background activity
   array     service  vdisks  pdisks  spares  threshold  BER      trim  free space  duration  task   progress  priority
-----------  -------  ------  ------  ------  ---------  -------  ----  ----------  --------  -------------------------
DA1          no            5      49    2,27          2  enable   no       339 TiB   14 days  inactive    56%  low

                                    declustered                        checksum
vdisk               RAID code           array     vdisk size  block size  granularity  state remarks
------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
RG001LOGHOME        4WayReplication     DA1             176 GiB      2 MiB      4096      ok    log
RG001VS002          3WayReplication     DA1            7148 GiB      2 MiB     32 KiB     ok
RG001VS001          8+2p                DA1              50 TiB     16 MiB     32 KiB     ok
RG001VS006          8+3p                DA1              63 GiB     16 MiB     32 KiB     ok
RG001VS005          8+3p                DA1              63 GiB     16 MiB     32 KiB     ok

config data         declustered array   spare space    remarks
------------------  ------------------  -------------  -------
rebuild space       DA1                 44 pdisk

config data         disk group fault tolerance         remarks
------------------  ---------------------------------  -------
rg descriptor       4 pdisk                            limiting fault tolerance
system index        4 pdisk                            limited by rg descriptor

vdisk               disk group fault tolerance         remarks
------------------  ---------------------------------  -------
RG001LOGHOME        3 pdisk
RG001VS002          2 pdisk
RG001VS001          2 pdisk
RG001VS006          3 pdisk
RG001VS005          3 pdisk

active recovery group server                    servers
----------------------------------------------  -------
c145f03n03.gpfs.net                             c145f03n03.gpfs.net,c145f03n04.gpfs.net
rebuild space       DA1                 44 pdisk

config data         disk group fault tolerance         remarks
------------------  ---------------------------------  -------
rg descriptor       4 pdisk                            limiting fault tolerance
system index        4 pdisk                            limited by rg descriptor

vdisk               disk group fault tolerance         remarks
------------------  ---------------------------------  -------
RG001LOGHOME        3 pdisk
RG001VS002          2 pdisk
RG001VS001          2 pdisk
RG001VS006          3 pdisk
RG001VS005          3 pdisk

active recovery group server                    servers
----------------------------------------------  -------
c145f03n03.gpfs.net                             c145f03n03.gpfs.net,c145f03n04.gpfs.net
```

For more information, see the following *IBM Storage Scale RAID: Administration* topic: “Determining pdisk-group fault-tolerance” on page 64.

4. The following example shows how to include pdisk information for BB01L:

```
mmlsrecoverygroup BB01L -L --pdisk
```

The system displays output similar to the following:

```
                   declustered                    current       allowable
recovery group       arrays     vdisks  pdisks  format version format version
-----------------  -----------  ------  ------  -------------- --------------
BB01L                       1       3      52  5.1.2.0        5.1.2.0

declustered   needs                            replace                 scrub            background activity
   array     service  vdisks  pdisks  spares  threshold  BER      trim  free space   duration  task   progress  priority
-----------  -------  ------  ------  ------  ---------  ----     -----  ----------  --------  ------------------------
DA1          no             3      52    2,27          2  enable   no       360 TiB     0 day   scrub       45%  low

                    n. active,   declustered              state,
pdisk               total paths     array     free space  remarks
-----------------   -----------  -----------  ----------  -------
e1s007                2,  4      DA1          7472 GiB    ok
e1s008                2,  4      DA1          7464 GiB    ok
e1s009                2,  4      DA1          7472 GiB    ok
e1s010                2,  4      DA1          7456 GiB    ok
.
.
.
active recovery group server                    servers
-----------------------------------------------  -------
c145f03n04.gpfs.net                              c145f03n04.gpfs.net,c145f03n03.gpfs.net
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmchrecoverygroup command” on page 285
- “mmcrrecoverygroup command” on page 288
- “mmdelrecoverygroup command” on page 301
- “mmlspdisk command” on page 324
- “mmlsrecoverygroupevents command” on page 335
- “mmlsvdisk command” on page 337.

Location

`/usr/lpp/mmfs/bin`

mmlsrecoverygroupevents command

Displays the IBM Storage Scale RAID recovery group event log.

Synopsis

```
mmlsrecoverygroupevents RecoveryGroupName [-T] [--days Days]
                        [--long-term Codes] [--short-term Codes]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmlsrecoverygroupevents` command displays the recovery group event log, internally divided into the following two logs:

short-term log
: Contains more detail than the long-term log, but due to space limitations may not extend far back in time

long-term log
Contains only brief summaries of important events and therefore extends much further back in time

Both logs use the following severity codes:

C
Commands (or configuration)

These messages record a history of commands that changed the specified recovery group.

E
Errors

W
Warnings

I
Informational messages

D
Details

By default, `mmlsrecoverygroupevents` displays both long-term and short-term logs merged together in order of message time stamp. Given the `--long-term` option, it displays only the requested severities from the long-term log. Given the `--short-term` option, it displays only the requested severities from the short-term log. Given both `--long-term` and `--short-term` options, it displays the requested severities from each log, merged by time stamp.

Note: The recovery group must be active to run this command.

Parameters

RecoveryGroupName
Specifies the name of the recovery group for which the event log is to be displayed.

-T
Indicates that the time is to be shown in decimal format.

--days *Days*
Specifies the number of days for which events are to be displayed.

For example, `--days 3` specifies that only the events of the last three days are to be displayed.

--long-term *Codes*
Specifies that only the indicated severity or severities from the long-term log are to be displayed. You can specify any combination of the severity codes listed in "Description" on page 335.

For example, `--long-term EW` specifies that only errors and warnings are to be displayed.

--short-term *Codes*
Specifies that only the indicated severity or severities from the short-term log are to be displayed. You can specify any combination of the severity codes listed in "Description" on page 335.

For example, `--short-term EW` specifies that only errors and warnings are to be displayed.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlsrecoverygroupevents` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

The following command example shows how to print the event logs of recovery group 000DE37BOT:

```
mmlsrecoverygroupevents 000DE37BOT
```

The system displays output similar to the following:

```
Mon May 23 12:17:36.916 2011 c08ap01 ST [I] Start scrubbing tracks of 000DE37BOTDA4META.
Mon May 23 12:17:36.914 2011 c08ap01 ST [I] Finish rebalance of DA DA4 in RG 000DE37BOT.
Mon May 23 12:13:00.033 2011 c08ap01 ST [D] Pdisk c109d4 of RG 000DE37BOT state changed from noRGD to ok.
Mon May 23 12:13:00.010 2011 c08ap01 ST [D] Pdisk c109d4 of RG 000DE37BOT state changed from noRGD/noVCD to noRGD.
Mon May 23 12:11:29.676 2011 c08ap01 ST [D] Pdisk c109d4 of RG 000DE37BOT state changed from noRGD/noVCD/noData to
noRGD/noVCD.
Mon May 23 12:11:29.672 2011 c08ap01 ST [I] Start rebalance of DA DA4 in RG 000DE37BOT.
Mon May 23 12:11:29.469 2011 c08ap01 ST [I] Finished repairing metadata in RG 000DE37BOT.
Mon May 23 12:11:29.409 2011 c08ap01 ST [I] Start repairing metadata in RG 000DE37BOT.
Mon May 23 12:11:29.404 2011 c08ap01 ST [I] Abort scrubbing tracks of 000DE37BOTDA4META.
Mon May 23 12:11:29.404 2011 c08ap01 ST [D] Pdisk c109d4 of RG 000DE37BOT state changed from missing/systemDrain/noRGD/
noVCD/noData/noPath
                                        to noRGD/noVCD/noData.
Mon May 23 12:11:29.401 2011 c08ap01 ST [D] Pdisk c109d4 of RG 000DE37BOT: path index 0 (/dev/rhdisk131): up.
Mon May 23 12:11:29.393 2011 c08ap01 ST [I] Path /dev/rhdisk131 of pdisk c109d4 reenabled.
Mon May 23 12:09:49.004 2011 c08ap01 ST [I] Start scrubbing tracks of 000DE37BOTDA4META.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmaddpdisk command" on page 268
- "mmchrecoverygroup command" on page 285
- "mmchcarrier command" on page 270
- "mmdelpdisk command" on page 299
- "mmdelvdisk command" on page 302
- "mmlspdisk command" on page 324
- "mmlsrecoverygroup command" on page 327
- "mmlsvdisk command" on page 337.

Location

`/usr/lpp/mmfs/bin`

mmlsvdisk command

Lists information for one or more IBM Storage Scale RAID vdisks.

Synopsis

```
mmlsvdisk [ --vdisk "VdiskName[;VdiskName...]" | --non-nsd ]
```

or

```
mmlsvdisk --recovery-group RecoveryGroupName [ --declustered-array DeclusteredArrayName ]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `mmlsvdisk` command lists information for one or more vdisks, specified various ways. Unless the `--recovery-group` option is specified, the information comes from the GPFS cluster configuration data.

Parameters

--vdisk "*VdiskName*[;*VdiskName*...]"
Specifies the name or names of the vdisk or vdisks for which the information is to be listed.

--non-nsd
Indicates that information is to be listed for the vdisks that are not associated with NSDs.

--recovery-group *RecoveryGroupName*
Specifies the name of the recovery group.

Note: The specified recovery group must be active to run this command.

--declustered-array *DeclusteredArrayName*
Specifies the name of the declustered array.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlsvdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

1. The following command example shows how to list all vdisks in the GPFS cluster:

```
mmlsvdisk
```

The system displays output similar to the following:

```
                                                    declustered  block size
vdisk name          RAID code        recovery group       array       in KiB    remarks
------------------  ---------------  ------------------  -----------  ----------  -------
000DE37BOTDA1DATA   8+3p             000DE37BOT          DA1                8192
000DE37BOTDA1META   4WayReplication  000DE37BOT          DA1                1024
000DE37BOTDA2DATA   8+3p             000DE37BOT          DA2                8192
000DE37BOTDA2META   4WayReplication  000DE37BOT          DA2                1024
000DE37BOTDA3DATA   8+3p             000DE37BOT          DA3                8192
000DE37BOTDA3META   4WayReplication  000DE37BOT          DA3                1024
000DE37BOTDA4DATA   8+3p             000DE37BOT          DA4                8192
000DE37BOTDA4META   4WayReplication  000DE37BOT          DA4                1024
000DE37BOTLOG       3WayReplication  000DE37BOT          LOG                1024  log
000DE37TOPDA1DATA   8+3p             000DE37TOP          DA1                8192
000DE37TOPDA1META   4WayReplication  000DE37TOP          DA1                1024
000DE37TOPDA2DATA   8+3p             000DE37TOP          DA2                8192
000DE37TOPDA2META   4WayReplication  000DE37TOP          DA2                1024
000DE37TOPDA3DATA   8+3p             000DE37TOP          DA3                8192
000DE37TOPDA3META   4WayReplication  000DE37TOP          DA3                1024
```

```
 000DE37TOPDA4DATA   8+3p             000DE37TOP          DA4                8192
 000DE37TOPDA4META   4WayReplication  000DE37TOP          DA4                1024
 000DE37TOPLOG       3WayReplication  000DE37TOP          LOG                1024  log
```

2. The following command example shows how to list only those vdisks in the cluster that do not have NSDs defined on them:

```
# mmlsvdisk --non-nsd
```

   The system displays output similar to the following:

```
                                                        declustered  block size
 vdisk name          RAID code        recovery group        array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 000DE37BOTLOG       3WayReplication  000DE37BOT          LOG                1024  log
 000DE37TOPLOG       3WayReplication  000DE37TOP          LOG                1024  log
```

3. The following command example shows how to see complete information about the vdisks in declustered array DA1 of recovery group 000DE37TOP:

```
mmlsvdisk --recovery-group 000DE37TOP --declustered-array DA1
```

   The system displays output similar to the following:

```
vdisk:
  name = "000DE37TOPDA1META"
  raidCode = "4WayReplication"
  recoveryGroup = "000DE37TOP"
  declusteredArray = "DA1"
  blockSizeInKib = 1024
  size = "250 GiB"
  state = "ok"
  remarks = ""

vdisk:
  name = "000DE37TOPDA1DATA"
  raidCode = "8+3p"
  recoveryGroup = "000DE37TOP"
  declusteredArray = "DA1"
  blockSizeInKib = 16384
  size = "17 TiB"
  state = "ok"
  remarks = ""
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmcrvdisk command” on page 291
- “mmdelvdisk command” on page 302
- “mmlspdisk command” on page 324
- “mmlsrecoverygroup command” on page 327
- “mmlsrecoverygroupevents command” on page 335.

Location

/usr/lpp/mmfs/bin

mmsyncdisplayid command

Synchronizes enclosure display IDs with the GPFS cluster configuration.

Synopsis

```
mmsyncdisplayid Enclosure | all
[ -N { nodeName[,nodeName...] | nodeFile | nodeClass } ]
[--dry-run] [ --log-level logLevel ]
```

Availability

Available with the Elastic Storage Server.

Description

The `mmsyncdisplayid` command synchronizes enclosure display IDs with the GPFS cluster configuration.

Parameters

***Enclosure* | `all`**
Specifies the name, serial number, or component ID of the enclosure. Or, specify `all` for all enclosures.

`-N` { *nodeName*[,*nodeName*...] | *nodeFile* | *nodeClass* }
Specifies the nodes that apply to the command. If `-N` is not specified, the target is all nodes in the cluster that have a server license.

`--dry-run`
Indicates that the changes resulting from the command are not to be recorded in the configuration file.

`--log-level` *logLevel*
Specifies the log level. Valid values are: critical, error, warning, info, detail, debug, and none.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmsyncdisplayid` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

To set the display ID to match the location of an enclosure as defined by the component configuration, and to see the result, enter the following commands:

```
mmsyncdisplayid all
mmlscomp --type storageenclosure
```

The system displays output similar to this:

```
    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  1818-80E     SV12345001     1818-80E-SV12345001  21
      2  1818-80E     SV12345007     1818-80E-SV12345007  01
      3  1818-80E     SV12345003     1818-80E-SV12345003  13
      4  1818-80E     SV12345005     1818-80E-SV12345005  05
      5  1818-80E     SV12345004     1818-80E-SV12345004  37
      6  1818-80E     SV12345002     1818-80E-SV12345002  25
      7  1818-80E     SV12345008     1818-80E-SV12345008  17
      8  1818-80E     SV12345006     1818-80E-SV12345006  33
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmaddcomp command” on page 264
- “mmaddcompspec command” on page 266
- “mmchcomp command” on page 273
- “mmchcomploc command” on page 275
- “mmchenclosure command” on page 277
- “mmchfirmware command” on page 279
- “mmdelcomp command” on page 295
- “mmdelcomploc command” on page 296
- “mmdelcompspec command” on page 298
- “mmdiscovercomp command” on page 304
- “mmlscomp command” on page 308
- “mmlscomploc command” on page 310
- “mmlscompspec command” on page 312
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320.

Location

`/usr/lpp/mmfs/bin`

mmvdisk command

Manages IBM Storage Scale RAID node classes, servers, recovery groups, vdisk sets, file systems, pdisks, vdisks, self-encrypting drives (SED), and NVMeOF target.

Synopsis

```
mmvdisk nodeclass  Verb [Parameters]
```

```
mmvdisk server  Verb [Parameters]
```

```
mmvdisk recoverygroup  Verb [Parameters]
```

```
mmvdisk vdiskset  Verb [Parameters]
```

```
mmvdisk filesystem  Verb [Parameters]
```

```
mmvdisk pdisk  Verb [Parameters]
```

```
mmvdisk vdisk  Verb [Parameters]
```

```
mmvdisk sed Verb [Parameters]
```

```
mmvdisk nvmeof Verb [Parameters]
```

Availability

Available on all IBM Storage Scale editions.

Description

The **mmvdisk** command is an integrated command suite and management methodology for IBM Storage Scale RAID. It greatly simplifies IBM Storage Scale RAID administration, and encourages and enforces consistent best practices with regard to server, recovery group, vdisk NSD, and file system configuration.

The **mmvdisk** command can be used to manage:

- The paired recovery groups of Elastic Storage Server, where a pair of servers divide a common set of disk enclosures into a pair of recovery groups, with each server taking exclusive primary responsibility for one recovery group of the pair.
- The scale-out recovery groups of IBM Storage Scale Erasure Code Edition, where a set of 4 to 32 identically equipped servers each take an equal share of responsibility in a single collective recovery group.
- The shared recovery groups of IBM Elastic Storage System 3000, where a single recovery group is defined on a common disk enclosure shared by two servers.
- The SED support by GNR to provide security for the data at rest.
- The NVMeOF target.

A recovery group that is managed by the **mmvdisk** command is called an mmvdisk recovery group.

New IBM Storage Scale clusters can use **mmvdisk** from the beginning to manage any IBM Storage Scale RAID recovery groups. The server and recovery group pairs of existing Elastic Storage Server installations can be converted to enable mmvdisk management. All existing non-mmvdisk recovery groups in a cluster must be converted to mmvdisk management before **mmvdisk** can be used to create new recovery groups in the cluster. The scale-out recovery groups of IBM Storage Scale Erasure Code Edition and the shared recovery groups of IBM Elastic Storage System 3000 are required to be managed by **mmvdisk**. If a cluster containing non-mmvdisk paired recovery groups wishes to add scale-out or shared recovery groups, the existing paired recovery groups must first be converted to mmvdisk management.

The **mmvdisk** command and the legacy IBM Storage Scale RAID administration commands are incompatible. When all recovery groups in a cluster are managed by the **mmvdisk** command, the legacy IBM Storage Scale RAID commands can not be used to create recovery groups or vdisks. Other legacy IBM Storage Scale RAID commands are similarly restricted. For example, if a recovery group is managed by **mmvdisk**, pdisk replacement must be performed using **mmvdisk pdisk replace**, since **mmchcarrier** will refuse to run against an mmvdisk recovery group. Conversely, **mmvdisk pdisk replace** will refuse to run against an existing recovery group that has not been converted to mmvdisk management, and **mmchcarrier** must be used with the unconverted recovery group.

The **mmvdisk** command structures IBM Storage Scale RAID around node classes, servers, recovery groups, vdisk sets, and file systems.

An mmvdisk node class contains the set of servers associated with a recovery group or recovery group pair. The IBM Storage Scale RAID configuration parameters for the servers are maintained under the node class. The node class also acts as a building block identifier for common server and disk capacity in the associated recovery group or recovery group pair.

The vdisks in a recovery group are sub-divided into log groups. A log group is a collection of vdisks that share a common RAID transaction log. In the case of a recovery group pair, there is just one log group per recovery group, and in this case a paired recovery group is trivially equivalent to its one and only log group. Each of the two paired recovery group servers has exclusive primary responsibility for the vdisks of one trivial log group. In the case of scale-out or shared recovery groups, the recovery group is sub-divided into multiple equal log groups, and each server is responsible for the vdisks of two log groups. Log groups act to equitably sub-divide and balance vdisks among the servers of a recovery group or recovery group pair.

The management of potentially large numbers of log group vdisks is simplified by treating the file system vdisk NSDs in the log groups of recovery groups collectively as members of vdisk sets. A vdisk set is a collection of vdisk NSDs with identical attributes from one or more recovery groups. Each log group of a recovery group included in a vdisk set contributes one member vdisk NSD. This means that when a vdisk set is defined using paired recovery groups, there will be one member vdisk NSD from each paired recovery group. When a vdisk set is defined using scale-out or shared recovery groups, there will be one member vdisk NSD from each log group of the recovery group, which is equivalent to two vdisk NSDs per server of the scale-out or shared recovery group.

A vdisk set must be managed as a unit: All member vdisk NSDs of a vdisk set must belong to the same mmvdisk file system. Defining a vdisk set across multiple recovery groups helps to ensure that a file system is balanced using identical vdisk NSDs from the recovery groups in the vdisk set.

An mmvdisk file system is an IBM Storage Scale RAID file system constructed from one or more vdisk sets. It is possible for an mmvdisk file system to contain non-vdisk NSDs, but the vdisk NSDs must all be members of vdisk sets, and non-vdisk and vdisk NSDs can not reside in the same file system storage pool.

In addition to the management structure provided by node classes, servers, recovery groups, vdisk sets, and file systems, **mmvdisk** also provides an interface to list and replace recovery group pdisks and to list individual vdisks.

Each mmvdisk sub-command has its own manual page with command syntax, information, and examples.

The **mmvdisk nodeclass** command manages mmvdisk recovery group server node classes.

The **mmvdisk server** command manages mmvdisk recovery group servers.

The **mmvdisk recoverygroup** command manages mmvdisk recovery groups.

The **mmvdisk vdiskset** command manages mmvdisk vdisk sets.

The **mmvdisk filesystem** command manages mmvdisk file systems.

The **mmvdisk pdisk** command manages mmvdisk recovery group pdisks.

The **mmvdisk vdisk** command lists individual recovery group vdisks.

The **mmvdisk sed** command manages self-encrypting drives.

The **mmvdisk nvmeof** command manages NVMeOF target.

Parameters

An **mmvdisk** command has the form **mmvdisk** *Noun Verb* [*Parameters*]. For example, **mmvdisk nodeclass list**, without any parameters, will list all the mmvdisk node classes in the cluster.

The nouns that mmvdisk recognizes are:

- **nodeclass:** Recovery group server node classes
- **server:** Recovery group servers within node classes
- **recoverygroup:** Recovery groups
- **vdiskset:** Vdisk sets across recovery groups
- **filesystem:** IBM Storage Scale file systems built from vdisk sets
- **pdisk:** Recovery group pdisks, for listing and maintenance
- **vdisk:** Individual vdisks for listing only

- **sed:** Self-encrypting drives
- **nvmeof:** NVMeOF target

The verbs are functions to be performed specific to a noun, and will always include `list`. Other possible verbs are: `create`, `add`, `delete`, and `change`.

After the noun and the verb, additional parameters might be required or optional, can occur in any order, and are always introduced with option flags. The vast majority of option flags are of the form `--option-name` and might or might not take following arguments.

A very few common traditional IBM Storage Scale RAID single-dash/single-character option flags can be used:

-N
This behaves like the traditional node specification.

-v
The traditional "verify" option to make sure objects are not already in use.

-p
The traditional "permanently damaged" option to delete inaccessible objects.

-L
The traditional "very long" detailed listing.

-Y
Provides colon-delimited raw output.

For convenience, some nouns can be abbreviated:

1. **`mmvdisk nc`** can be used in place of **`mmvdisk nodeclass`**.
2. **`mmvdisk rg`** can be used in place of **`mmvdisk recoverygroup`**.
3. **`mmvdisk vs`** can be used in place of **`mmvdisk vdiskset`**.
4. **`mmvdisk fs`** can be used in place of **`mmvdisk filesystem`**.

Some common option flags can also be abbreviated:

1. `--nc` can be used in place of `--node-class`.
2. `--rg` can be used in place of `--recovery-group`.
3. `--da` can be used in place of `--declustered-array`.
4. `--vs` can be used in place of `--vdisk-set`.
5. `--fs` can be used in place of `--file-system`.
6. `--bs` can be used in place of `--block-size`.
7. `--ss` can be used in place of `--set-size`.
8. `--usage` can be used in place of `--nsd-usage`.
9. `--pool` can be used in place of `--storage-pool`.
10. `--ft` can be used in place of `--fault-tolerance`.
11. `--lg` can be used in place of `--log-group`.

Depending on the context, when it is possible to list multiple comma-separated objects as the argument to an option flag, the keyword `all` can often be used as shorthand for listing all objects of a type. For example, `--vs all` can be used to specify all vdisk sets. The individual command usage statements will indicate this with a `--vdisk-set` {`all` | *VsName* [,*VsName*...]} syntax diagram.

When a command will delete (or undefine, in the case of vdisk sets) an object permanently, **`mmvdisk`** will warn the user by prompting for confirmation. In these cases, the prompt can be bypassed and answered in the affirmative with the `--confirm` option.

`mmvdisk` will also warn the user and prompt for confirmation before performing potentially long-running commands. There is no option to bypass the prompt in the case of long-running commands.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

1. To see all mmvdisk node classes in the cluster:

```
mmvdisk nodeclass list
```

   The system displays output similar to the following:

```
node class  recovery groups
----------  ---------------
ESS01       ESS01L, ESS01R
ESS02       ESS02L, ESS02R
```

2. To see all recovery group servers in the cluster:

```
mmvdisk server list
```

   The system displays output similar to the following:

```
 node
number  server                          node class  recovery groups  remarks
------  ------------------------------  ----------  ---------------  -------
     1  ess01io1.gpfs.net               ESS01       ESS01L, ESS01R
     2  ess01io2.gpfs.net               ESS01       ESS01L, ESS01R
     3  ess02io1.gpfs.net               ESS02       ESS02L, ESS02R
     4  ess02io2.gpfs.net               ESS02       ESS02L, ESS02R
```

3. To see all recovery groups in the cluster:

```
mmvdisk recoverygroup list
```

   The system displays output similar to the following:

```
                                                        needs    user
recovery group  active   current or master server      service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
ESS01L          yes      ess01io1.gpfs.net                no            2
ESS01R          yes      ess01io2.gpfs.net                no            2
ESS02L          yes      ess02io1.gpfs.net                no            2
ESS02R          yes      ess02io2.gpfs.net                no            2
```

4. To see all vdisk sets in the cluster:

```
mmvdisk vdiskset list
```

   The system displays output similar to the following:

```
vdisk set         created  file system  recovery groups
----------------  -------  -----------  ---------------
VS1               yes      FS1          ESS01L, ESS01R, ESS02L, ESS02R
VS2               yes      FS2          ESS01L, ESS01R, ESS02L, ESS02R
```

5. To see all file systems in the cluster:

```
mmvdisk filesystem list
```

The system displays output similar to the following:

```
file system  vdisk sets
-----------  ----------
FS1          VS1
FS2          VS2
```

Note: See the specific **mmvdisk** sub-commands for more detailed examples.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmvdisk nodeclass command” on page 346
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk pdisk command” on page 388
- “mmvdisk sed command” on page 394
- “mmvdisk nvmeof command” on page 397

Location

`/usr/lpp/mmfs/bin`

mmvdisk nodeclass command

Manages mmvdisk recovery group server node classes for IBM Storage Scale RAID.

Synopsis

```
mmvdisk nodeclass create --node-class NcName -N Node[,Node...]
```

or

```
mmvdisk nodeclass add --node-class NcName -N Node[,Node...]
```

or

```
mmvdisk nodeclass delete --node-class NcName -N Node[,Node...]
```

or

```
mmvdisk nodeclass delete --node-class NcName [--confirm]
```

or

```
mmvdisk nodeclass list [--node-class {all | NcName[,NcName...]}] [--custom-config] [-Y]
```

or

```
mmvdisk nodeclass change --node-class NcName --set-custom-config [Attribute[,Attribute...]]
```

or

```
mmvdisk nodeclass change --node-class NcName --add-custom-config [Attribute[,Attribute...]]
```

or

```
mmvdisk nodeclass change --node-class NcName --delete-custom-config [Attribute[,Attribute...] |
all]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk nodeclass** command to manage mmvdisk recovery group server node classes.

An mmvdisk node class is an IBM Storage Scale node class that can only be created, modified, or deleted through the **mmvdisk** command. Each IBM Storage Scale RAID recovery group managed by **mmvdisk** must be associated with an mmvdisk node class that contains only the servers for that recovery group.

The **mmvdisk** command uses the node class for a recovery group or recovery group pair to maintain the IBM Storage Scale RAID configuration settings for the recovery group and its servers. The node class is also used as a building block identifier for the common server and disk resources required to store file system data in the associated recovery group or recovery group pair.

The association between an mmvdisk node class and an mmvdisk recovery group can be established in two ways:

1. The **mmvdisk nodeclass create** command is used to create a node class containing the intended servers for the recovery group, and the **mmvdisk recoverygroup create** command is used to create an mmvdisk recovery group using the node class.
2. The **mmvdisk recoverygroup convert** command is used to convert two paired recovery groups and their two servers to mmvdisk management. The **mmvdisk recoverygroup convert** command must be supplied with a node class name to be used as the mmvdisk node class for the converted recovery group pair.

The **mmvdisk nodeclass add** and **mmvdisk nodeclass delete** commands can be used to modify or delete an mmvdisk node class that has not yet been configured to be IBM Storage Scale RAID recovery group servers. This can be useful for correcting the member servers of an unconfigured mmvdisk node class.

The **mmvdisk nodeclass change** commands can be used to modify the customized configuration attributes of an mmvdisk node class. The `--set-custom-config` option sets customized configuration attributes. All original customized configuration attributes are overwritten. The `--add-custom-config` option adds customized configuration attributes to the original customized configuration attributes. The `--delete-custom-config` option deletes customized configuration attributes from the original customized configuration attributes. If all is specified, it deletes all customized configuration attributes.

When a recovery group is associated with an mmvdisk node class, changes to the node class can only be made using the **mmvdisk recoverygroup** command. For example, the **mmvdisk recoverygroup add --recovery-group** *RG01* `-N` *server11* command will add the new server node both to the recovery group and to the mmvdisk node class for the recovery group.

To change an mmvdisk node class that is configured but not yet associated with a recovery group, the node class must first be unconfigured using the **mmvdisk server unconfigure** command.

The **mmvdisk nodeclass list** command can be used to list all mmvdisk node classes and show the associated recovery groups, or to also list the member servers of mmvdisk node classes. When the `--custom-config` option is specified, it shows all customized configuration attributes for node classes.

Parameters

mmvdisk nodeclass create
Create a new mmvdisk node class. The node class must not already exist.

mmvdisk nodeclass add
Add server nodes to an existing mmvdisk node class. The node class must not be configured or associated with a recovery group. The server nodes must not already belong to an mmvdisk node class.

mmvdisk nodeclass delete
Delete server nodes from an existing mmvdisk node class, or delete an entire mmvdisk node class. The node class must not be configured or associated with a recovery group..

mmvdisk nodeclass list
List all the mmvdisk node classes in an IBM Storage Scale cluster, or list the member servers in the specified mmvdisk node classes.

mmvdisk nodeclass change
Changes the mmvdisk node classes with customized configuration in an IBM Storage Scale cluster.

--node-class ***NcName***
Specifies a single mmvdisk node class name to be either created, deleted, added to, or deleted from.

--node-class {all | *NcName*[,*NcName*...]}
For the **mmvdisk nodeclass list** command, specifies the mmvdisk node class names for which the member servers are also to be shown.

-N ***Node*[,*Node*...]**
Specifies the IBM Storage Scale cluster nodes with which to create an mmvdisk node class, or the list of cluster nodes to be added to or deleted from an mmvdisk node class.

--confirm
Confirms the deletion of an entire mmvdisk node class. This flag bypasses the interactive prompt that would otherwise be printed to confirm node class deletion.

--custom-config [Attribute[,Attribute...]]
Shows customized configuration attributes for the node class.

--set-custom-config [Attribute[,Attribute...]]
Sets customized configuration attributes for the node class that will not be checked by **mmvdisk server** command.

--add-custom-config [Attribute[,Attribute...]]
Adds customized configuration attributes for the node class that will not be checked by **mmvdisk server** command.

--delete-custom-config [Attribute[,Attribute...]]
Deletes customized configuration attributes for the node class that will not be checked by **mmvdisk server** command.

-Y
Specifies that the **mmvdisk nodeclass list** command produce colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk nodeclass** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

- To create an mmvdisk node class named ESS01 containing nodes server01 and server02:

```
mmvdisk nodeclass create --node-class ESS01 -N server01,server02
```

The system displays output similar to the following:

```
mmvdisk: Node class 'ESS01' created.
```

- To list the mmvdisk node classes in a cluster:

```
mmvdisk nodeclass list
```

The system displays output similar to the following:

```
node class  recovery groups
----------  --------------
ESS01       ESS01L, ESS01R
ESS02       ESS02L, ESS02R
```

- To list all the mmvdisk node classes in a cluster and all their member nodes:

```
mmvdisk nodeclass list --node-class all
```

The system displays output similar to the following:

```
node class  recovery groups  member nodes
----------  --------------  ------------
NC01        RG01            s01, s02, s03, s04, s05, s06, s07, s08, s09, s10
NC02        RG02            s11, s12, s13, s14, s15, s16, s17, s18, s19, s20
```

- To list the mmvdisk node classes in a cluster by using the `--custom-config` option, issue the following command:

```
mmvdisk nodeclass list --custom-config
```

The system displays output similar to the following:

```
node class  recovery groups   custom config
----------  --------------   -------------
ESS01       ESS01L, ESS01R    nsdRAIDSmallVdiskThreshold
ESS02       ESS02L, ESS02R    workerThreads, numaMemoryInterleave
```

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk pdisk command” on page 388

Location

`/usr/lpp/mmfs/bin`

mmvdisk server command

Manages mmvdisk recovery group servers for IBM Storage Scale RAID.

Synopsis

```
mmvdisk server configure {--node-class NcName | -N Node[,Node...]}
                         [--maxblocksize {2M | 4M | 8M | 16M}]
                         [--pagepool {nM |nG | n%}]
                         [--custom-config Attribute[=value][,Attribute[=value]...]]
                         [--update]
                         [--recycle {none |one |all | Number}]
```

or

```
mmvdisk server configure --node-class NcName --verify [-Y]
```

or

```
mmvdisk server unconfigure {--node-class NcName | -N Node[,Node...]}
                           [--recycle {none | one | all | Number}]
```

or

```
mmvdisk server change --node-class NcName
                     {[--pagepool {nM | nG | n%}] [--vdisk-space n%] [--map-memory n%]}
                     [--recycle {none | one | all | Number}]
```

or

```
mmvdisk server list [-Y]
```

or

```
mmvdisk server list {--node-class NcName |-N Node[,Node...] | --recovery-group RgName}
                    [{[--version][--config][--disk-topology ][--disk-list DiskExpr]} [--fanout
N]] [-Y]
```

or

```
mmvdisk server list -N Node --disk-topology {[--disk-topology][--disk-list DiskExpr]} -L  [-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk server** command to manage mmvdisk recovery group servers.

Before an IBM Storage Scale RAID recovery group can be created, the mmvdisk node class for the intended recovery group or recovery group pair must be configured to run IBM Storage Scale RAID and the IBM Storage Scale daemon must be restarted on each server in the node class. The **mmvdisk server configure** command examines the nodes in an mmvdisk node class, verifies that the node class members meet the same memory and disk topology requirements, and sets the appropriate configuration parameters. The **mmvdisk server configure** command can be used to optionally recycle the IBM Storage Scale daemon on the node class to ensure that the new configuration is in effect.

The configuration parameters set by **mmvdisk server configure** depend on the amount of server memory and the type of server disk topology. All node class members must have, within ten percent, the

same amount of memory, and all node class members must have the same general type of server disk topology: either a paired recovery group (Elastic Storage Server) disk topology or a scale-out recovery group (IBM Storage Scale Erasure Code Edition) disk topology or a shared recovery group (IBM Elastic Storage System 3000) disk topology. The **mmvdisk server configure** command will not configure a node class where the server memory differs by more than ten percent or where a mixture of recovery group topologies is present.

The **mmvdisk server configure** command can be used to optionally recycle the IBM Storage Scale daemon on the recovery group server node class to ensure that the new configuration is in effect for a subsequent **mmvdisk recoverygroup create** command.

The **mmvdisk server configure** command can be used with the `--update` option on a node class where a recovery group is already present. This can be useful when an updated version of the **mmvdisk** command uses new configuration parameters, or when the server memory or disk topology has changed and the configuration parameters based on memory or the number of disks should be recalculated. The `--recycle` option can be used with `--update` to make the updated configuration parameters take effect by restarting the IBM Storage Scale daemon on the servers in the node class.

The **mmvdisk server configure** command can also be used to configure individual nodes for IBM Storage Scale RAID to prepare for maintenance use cases where servers are replaced or added to a recovery group and its node class.

The `--verify` option of the **mmvdisk server configure** command verifies whether or not the configuration settings for an mmvdisk server node class are as expected. The `--verify` option will check the real memory and server disk topology for each node in the node class, and check whether the IBM Storage Scale RAID configuration attributes for the node class are set to the expected values for the memory and topology. The **mmvdisk server configure --verify** command can be used to check whether a new release of IBM Storage Scale RAID has updated best practice configuration values; to check whether configuration values are affected by changes that have been made to server memory or server disk topology; or to check whether accidental changes have been made to the node class configuration. The **mmvdisk** command bypasses the check for the attributes that belong to customized configuration attributes of node class. The customized configuration attributes might be set by the `--custom-config` option or the **mmvdisk nodeclass change** command. If an unexpected configuration is reported, the **mmvdisk server configure --update** command can be used to reset the node class to the correct configuration values.

The **mmvdisk server unconfigure** command will remove IBM Storage Scale RAID configuration from a node class where a recovery group was either deleted or never created. The **mmvdisk server unconfigure** command can also be used to unconfigure IBM Storage Scale RAID on individual nodes for maintenance use cases where servers are replaced or deleted from a recovery group and its node class.

Use the **mmvdisk server change** command to safely change the server memory configuration of an mmvdisk server node class. The memory configuration of a node class affects the number and size of vdisk sets that can be defined in the recovery groups of the node class. The **mmvdisk server change** command will not allow changes that reduce the available memory below what is required by the vdisk set definitions in the node class. The three options that affect node class server memory for vdisk sets are:

1. The `--pagepool` option, which adjusts the node class IBM Storage Scale pagepool attribute.
2. The `--vdisk-space` option, which adjusts the node class nsdRAIDBufferPoolSizePct attribute for the percentage of pagepool dedicated to vdisks and IBM Storage Scale RAID.
3. The `--map-memory` option, which adjusts the node class nsdRAIDNonStealableBufPct attribute for the percentage of vdisk space dedicated to the track address maps for each vdisk.

The **mmvdisk server change** command should only very rarely be necessary. The default values for the server memory configuration options should be suitable for most circumstances. The pagepool default is dependent on the real server memory and the recovery group type in the node class. The vdisk space default is always 80% of pagepool, and the map memory default is always 50% of the vdisk space. In the event that, for example, the map memory needs to be increased to address very large vdisks with very small block sizes, the **mmvdisk server change --node-class NC --map-memory 60%** command can be used to safely adjust the server memory configuration in node class `NC`.

The `--recycle one` option can be used with the **mmvdisk server change** command to make any changes take effect by restarting IBM Storage Scale one node at a time in the node class.

The **mmvdisk server list** command, without any options, will list all the configured IBM Storage Scale RAID servers in the cluster. It identifies all recovery group servers and all nodes configured to be recovery group servers, regardless of whether the servers are members of an mmvdisk node class. This can be useful for planning server maintenance and planning the conversion of recovery groups and servers to mmvdisk management.

The **mmvdisk server list** command supports several options useful for showing server characteristics. The `--node-class` option will show whether the servers in a node class are active and the recovery group they serve. The `--version` option can be used to show the IBM Storage Scale and operating system versions for servers. The `--config` option can be used to show important aspects of the IBM Storage Scale RAID server configuration.

The **mmvdisk server list** `--disk-topology` option is important for reporting the supported IBM Storage Scale RAID disk topology on a node class or server. It should be used before **mmvdisk server configure** on a node class, to verify that all the node class members have the same server disk topology. It can also be used to help identify disk configuration problems that prevent nodes from acting as recovery group servers.

The **mmvdisk server list** `--disk-list` *diskExpr* option is important for reporting IBM Storage Scale RAID disk list on a node class or server. *diskExpr* is a subset of disk list which might be useful if you do not want to use all disks on a node to create recovery group.

Parameters

mmvdisk server configure
: Configure nodes or an mmvdisk node class to be IBM Storage Scale RAID servers. The nodes or node class must not already be serving a recovery group or already be configured, except when the `--update` option is used for a node class where a recovery group is present.

mmvdisk server unconfigure
: Unconfigure nodes or an mmvdisk node class to no longer be IBM Storage Scale RAID servers. The nodes or node class must no longer be associated with a recovery group.

mmvdisk server change
: Change server memory configuration attributes for an mmvdisk server node class. The pagepool, vdisk buffer space, and vdisk map memory can be adjusted for a node class. The **mmvdisk server change** command will not allow changes that reduce the available memory below what is required for the vdisk set definitions in the node class.

mmvdisk server list
: List IBM Storage Scale RAID servers and their characteristics. A node is considered to be a server if it is in the server list of a recovery group, or if it is member of an mmvdisk node class, or if it has the IBM Storage Scale RAID nsdRAIDTracks attribute set.

--node-class ***NcName***
: Specifies a single mmvdisk node class name to be configured, unconfigured, or listed.

-N ***Node*****[,*****Node*****...]**
: Specifies individual nodes to be configured, unconfigured, or listed.

--recycle {none | one | all | ***Number*****}**
: With **mmvdisk server configure** or **mmvdisk server unconfigure**, specifies the number of nodes to be simultaneously restarted so that IBM Storage Scale configuration changes can take effect. The default is `none`, meaning that the administrator is responsible for using the **mmshutdown** and **mmstartup** commands to recycle the nodes or node class. The keyword `one` will recycle one node at a time, the keyword `all` will recycle all specified nodes simultaneously, or a *Number* of nodes can be chosen to be recycled at the same time. Care should be taken when recycling nodes if it is desired to maintain quorum availability in the IBM Storage Scale cluster.

--pagepool {*n*M | *n*G | *n*%}
With **mmvdisk server configure** and **mmvdisk server change**, choose a specific IBM Storage Scale pagepool value. Normally, **mmvdisk** chooses the best appropriate pagepool size for the real memory and disk topology in the server node class, so this option should only be used under instruction from IBM.

--maxblocksize {2M | 4M | 8M | 16M}
With **mmvdisk server configure**, also set the IBM Storage Scale cluster `maxblocksize` value. Even though `maxblocksize` is not a server-specific configuration value, IBM Storage Scale RAID is often used with large file system block sizes, and this option provides a convenient way to set the maximum file system block size. This would typically be done only when configuring the first recovery group in a cluster, and it requires that the administrator ensure that the IBM Storage Scale daemon on all cluster nodes is recycled (not only on the recovery group server nodes).

--custom-config *Attribute*[=*value*][,*Attribute*[=*value*]...]
With **mmvdisk server configure**, also set the IBM Storage Scale cluster `customized configuration` value. When mmvdisk (*See IBM Storage Scale RAID: Administration*) executes, it calls **mmchconfig** to set the value of attributes and add these attributes to node class's customized configuration attributes.

--update
With **mmvdisk server configure** for a node class that already has a recovery group present, update the node class configuration parameters according to the current **mmvdisk** version and current server memory and disk topology.

--verify
With the **mmvdisk server configure** command, verify whether or not the IBM Storage Scale RAID configuration attributes for an mmvdisk server node class are set to the expected values.

--vdisk-space *n*%
With the **mmvdisk server change** command, specify the percentage of pagepool that should be reserved as space for vdisks and IBM Storage Scale RAID. This corresponds to the nsdRAIDBufferPoolSizePct configuration attribute, and the default value is 80%. Only in rare circumstances should this value need to be changed, so this option should only be used under instruction from IBM.

--map-memory *n*%
With the **mmvdisk server change** command, specify the percentage of the vdisk space that should be reserved to map the track addresses of vdisks. This corresponds to the nsdRAIDNonStealableBufPct configuration attribute, and the default value is 50%. Only in rare circumstances should this value need to be changed, so this option should only be used under instruction from IBM.

--recovery-group *RgName*
With **mmvdisk server list**, lists the servers for the named recovery group.

--version
With **mmvdisk server list**, lists the IBM Storage Scale and operating system version for the servers.

--config
With **mmvdisk server list**, lists the IBM Storage Scale RAID configuration for the servers.

--disk-topology
With **mmvdisk server list**, lists the discovered IBM Storage Scale RAID disk topology on the servers.

--disk-list *DiskExpr*
With **mmvdisk server list**, lists the discovered IBM Storage Scale RAID disks on the servers. If *DiskExpr* is specified, mmvdisk shows only the list of disks that belong to *DiskExpr*. For more information, see "mmvdisk recoverygroup command" on page 357.

--fanout *N*
With **mmvdisk server list** `--disk-topology`, chooses how many servers to query simultaneously for their disk topologies. The default is 32. A `fanout` of 1 can be useful when disk fabric errors with twin-tailed server pairs are suspected.

-L
With **mmvdisk server list** --disk-topology and a single node, provide a detailed listing of the discovered IBM Storage Scale RAID disk topology. This is very useful when disk topology errors are suspected.

-Y
Specifies that the **mmvdisk server list** and **mmvdisk server configure --verify** commands produce colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk server** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

- To display the IBM Storage Scale RAID disk topology of the servers in mmvdisk node class, ESS01:

```
mmvdisk server list --node-class ESS01 --disk-topology
```

The system displays output similar to the following:

```
 node                                 needs    matching
number  server                       attention   metric   disk topology
------  -------------------------------------  ---------  --------  -------------
     1  server01.gpfs.net            no          100/100  ESS GL6
     2  server02.gpfs.net            no          100/100  ESS GL6
```

- To display the IBM Storage Scale RAID disk topology and disk list of the servers in mmvdisk node class, nc3500:

```
mmvdisk server list --node-class nc3500 --disk-topology --disk-list
```

The system displays output similar to the following:

```
 node                                 needs    matching
number  server                       attention   metric   disk topology
------  -------------------------------------  ---------  --------  -------------
     2  c145f11san08a                no          100/100  ESS 3500 FN2 24 NVMe
     1  c145f11san08b                no          100/100  ESS 3500 FN2 24 NVMe

                                    disk          list of
type  size(Mb)  spin   server         count       disk/location
----  --------  -----  -------------  -----  ------------------------
NVMe  3662830   0      c145f11san08a  24     nvme[0-23]n1 / 78E4001-[1-24]
                       c145f11san08b  24     nvme[0-23]n1 / 78E4001-[1-24]
```

- To configure mmvdisk node class ESS01 to be IBM Storage Scale RAID recovery group servers and restart the daemons one at a time to make the configuration take effect:

```
mmvdisk server configure --node-class ESS01 --recycle one
```

The system displays output similar to the following:

```
mmvdisk: Checking resources for specified nodes.
```

```
mmvdisk: Node 'server01.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node 'server02.gpfs.net' has a paired recovery group disk topology.
mmvdisk: Node class 'ESS01' has a paired recovery group disk topology.
mmvdisk: Setting configuration for node class 'ESS01'.
mmvdisk: Node class 'ESS01' is now configured to be recovery group servers.
mmvdisk: Restarting GPFS on the following nodes:
mmvdisk:     server01.gpfs.net
mmvdisk: Restarting GPFS on the following nodes:
mmvdisk:     server02.gpfs.net
```

- To show the status of the newly-configured mmvdisk node class ESS01:

```
mmvdisk server list --node-class ESS01
```

The system displays output similar to the following:

```
 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  server01.gpfs.net               yes      configured, idle
     2  server02.gpfs.net               yes      configured, idle
```

This shows that the servers are configured and running, but IBM Storage Scale RAID is idle because recovery groups have not yet been created.

- To display the disk topology of the scale-out recovery group servers of recovery group RG01:

```
mmvdisk server list --recovery-group RG01 --disk-topology
```

The system displays output similar to the following:

```
 node                                 needs    matching
number  server                       attention   metric   disk topology
------  -------------------------------  ---------  --------  -------------
     1  s01                           no          100/100  ECE 2 SSD/NVMe and 5 HDD
     2  s02                           no          100/100  ECE 2 SSD/NVMe and 5 HDD
     3  s03                           no          100/100  ECE 2 SSD/NVMe and 5 HDD
     4  s04                           no          100/100  ECE 2 SSD/NVMe and 5 HDD
     5  s05                           no          100/100  ECE 2 SSD/NVMe and 5 HDD
```

- To display the current status of the servers of recovery group RG01:

```
mmvdisk server list --recovery-group RG01
```

The system displays output similar to the following:

```
 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  s01                           yes      serving RG01: LG003, LG008
     2  s02                           yes      serving RG01: root, LG005, LG010
     3  s03                           yes      serving RG01: LG002, LG007
     4  s04                           yes      serving RG01: LG001, LG006
     5  s05                           yes      serving RG01: LG004, LG009
```

- In an mmvdisk server node class where vdisk sets are defined, a vdisk space of 20% and a map memory of 30% will almost certainly be a memory configuration error, and the **mmvdisk server change** command will not allow it:

```
mmvdisk server change --node-class ESS --vdisk-space 20% --map-memory 30%
```

The system displays output similar to the following:

```
mmvdisk: Checking resources for specified nodes.
mmvdisk: The requested change to node class 'ESS':
mmvdisk:   --vdisk-space 20% (config attribute 'nsdRAIDBufferPoolSizePct')
mmvdisk:   --map-memory 30% (config attribute 'nsdRAIDNonStealableBufPct')
mmvdisk: results in an available vdisk set map memory of 4558 MiB.
mmvdisk: This is less than the required vdisk set map memory of 26 GiB
```

```
mmvdisk: based on the current vdisk set definitions in the node class.
mmvdisk: Command failed. Examine previous error messages to determine cause.
```

This example is shown only as a reference. Please be reminded that the **mmvdisk server change** command should almost never be necessary, since the default values for the server memory configuration options are suitable for most circumstances.

- To verify that the mmvdisk server node class ESS is properly configured:

```
mmvdisk server configure --node-class ESS --verify
```

The system displays output similar to the following:

```
mmvdisk: Checking resources for specified nodes.
mmvdisk: Node class 'ESS' has 123 GiB total real memory per server.
mmvdisk: Node class 'ESS' has a paired recovery group disk topology.
mmvdisk: Node class 'ESS' has server disk topology 'ESS GL6'.
mmvdisk: Node class 'ESS' uses 'default.paired' recovery group configuration.

daemon configuration attribute      expected value  configured value
----------------------------------  --------------  ----------------
pagepool                            79665522278     as expected
nsdRAIDTracks                       128K            as expected
nsdRAIDBufferPoolSizePct            80              as expected
nsdRAIDNonStealableBufPct           50              as expected
pagepoolMaxPhysMemPct               90              as expected
nspdQueues                          64              as expected
nsdRAIDBlockDeviceMaxSectorsKB      0               as expected
nsdRAIDBlockDeviceNrRequests        0               as expected
nsdRAIDBlockDeviceQueueDepth        0               as expected
nsdRAIDBlockDeviceScheduler         off             as expected
nsdRAIDFlusherFWLogHighWatermarkMB  1000            as expected
nsdRAIDMaxPdiskQueueDepth           128             as expected
nsdRAIDSmallThreadRatio             2               as expected
nsdRAIDThreadsPerQueue              16              as expected
ignorePrefetchLUNCount              yes             as expected
maxBufferDescs                      2m              as expected
maxFilesToCache                     128k            as expected
maxMBpS                             30000           as expected
maxStatCache                        128k            as expected
nsdMaxWorkerThreads                 3842            as expected
nsdMinWorkerThreads                 3842            as expected
nsdMultiQueue                       256             as expected
numaMemoryInterleave                yes             as expected
panicOnIOHang                       yes             as expected
pitWorkerThreadsPerNode             32              as expected
workerThreads                       1024            as expected

mmvdisk: All configuration attribute values are as expected or customized.
```

See also

- “mmvdisk command” on page 341
- “mmvdisk nodeclass command” on page 346
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk pdisk command” on page 388

Location

/usr/lpp/mmfs/bin

mmvdisk recoverygroup command

Manages mmvdisk recovery groups for IBM Storage Scale RAID.

Synopsis

or

```
mmvdisk recoverygroup create  --recovery-group RgName
                             --node-class NcName [--match  N]
                             [--log-home-da DaName] [-v {yes | no}]
                             [--fanout N] [--defer-log-format]
                             [--trim-da DaType[,DaType...]]
                             [--split-da [DaType:]N[,DaType:N...]]
                             [--disk-list DiskExpr]
                             [--ignore-disk-list DiskExpr]
```

or

```
mmvdisk recoverygroup create --complete-log-format
```

or

```
mmvdisk recoverygroup convert --recovery-group RgName1[,RgName2]
                              --node-class NcName
                              [--inherit-config {yes | no [--pagepool {nM | nG | n%}]
                              [--recycle {none | one | all | Number}]}]
                              [--dry-run]
```

or

```
mmvdisk recoverygroup resize --recovery-group RgName1[,RgName2]
                             [--split-da [daType:]N[,daType:N...]]
                             [--disk-list DiskExpr]
                             [--ignore-disk-list DiskExpr]
                             [-v {yes | no}] [--fanout N]
                             [--dry-run]
```

or

```
mmvdisk recoverygroup add  --recovery-group RgName
                             -N Node[,Node...]
                             [--disk-list DiskExpr]
                             [--ignore-disk-list DiskExpr]
                             [--match  N] [--fanout N] [-v {yes |
no}]
```

or

```
mmvdisk recoverygroup add --recovery-group RgName --complete-node-add
```

or

```
mmvdisk recoverygroup replace  --recovery-group RgName
                             -N Node --new-node NewNode [--match N]
                             [--disk-list DiskExpr]
                             [--ignore-disk-list DiskExpr]
                             [--fanout N] [-v {yes | no}]
```

or

```
mmvdisk recoverygroup delete  --recovery-group RgName [-N Node[,Node...]]
                             [--confirm] [-p] [--sanitize]
```

or

```
mmvdisk recoverygroup change --recovery-group RgName
                             {--primary Node [--backup Node] [-v {yes | no}] |
                             --active {Node | DEFAULT} |
                             --version {VersionString | LATEST} |
                             --declustered-array DaName {[--spares NumberOfSpares]
                             [--vcd-spares NumberOfVcdSpares] [--scrub-duration NumberOfDays]
                             [--replace-threshold NumberOfPdisks]
                             [--bit-error-rate {enable | disable}]
                             [--trim-da {yes | no}]} |
                             --declustered-array {all | DaName[,DaName...]}
                             {--spare-nodes {DEFAULT | N} | --spare-pdisks {DEFAULT | N}} |
                             --suspend -N Node [--window N]|
                             --resume -N Node |
                             --restart |
                             --refresh-pdisk-info}
```

or

```
mmvdisk recoverygroup list [-Y]
```

or

```
mmvdisk recoverygroup list --recovery-group RgName
                           [--version] [--server] [--declustered-array] [--pdisk]
                           [--log-group] [--vdisk-set] [--vdisk] [--fault-tolerance] [-Y]
```

or

```
mmvdisk recoverygroup list --recovery-group RgName  --all [-Y]
```

or

```
mmvdisk recoverygroup list  --declustered-array [-Y]
```

or

```
mmvdisk recoverygroup list  --not-ok [-Y]
```

or

```
mmvdisk recoverygroup list --recovery-group RgName  --events
                           [--days Days] [--decimal]
                           [--long-term Codes] [--short-term Codes]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk recoverygroup** command to manage IBM Storage Scale RAID recovery groups.

The **mmvdisk recoverygroup** command will:

- Create new mmvdisk paired recovery groups (Elastic Storage Server).
- Create a new mmvdisk scale-out recovery group (IBM Storage Scale Erasure Code Edition).
- Create a new mmvdisk shared recovery group (IBM Elastic Storage System 3000).
- Convert existing non-mmvdisk paired recovery groups to run under mmvdisk management.
- Resize existing mmvdisk recovery groups to a supported disk topology upgrade.
- Add servers to an mmvdisk scale-out recovery group.

- Replace a server in an mmvdisk scale-out recovery group.
- Delete servers from an mmvdisk scale-out recovery group.
- Change the attributes of mmvdisk recovery groups.
- List recovery group information.
- Delete mmvdisk recovery groups.

Use the **mmvdisk recoverygroup create** command to create an IBM Storage Scale RAID recovery group pair (Elastic Storage Server). This requires an mmvdisk node class containing the two servers. The node class must already have been configured using the **mmvdisk server configure** command, and the servers must have been restarted with the configuration in effect. The two servers must have identical IBM Storage Scale RAID disk topologies. The names of the two paired recovery groups must be supplied using the `--recovery-group` *RgName1*,*RgName2* parameter. The primary and backup servers for the two recovery groups are determined by a case-insensitive sort of the two server names. The first supplied recovery group will get the server that sorts first as primary and the server that sorts second as backup; the second supplied recovery group will get the server that sorts second as primary and the server that sorts first as backup. If desired, these assignments can be changed later using the **mmvdisk recoverygroup change** command.

Use the **mmvdisk recoverygroup create** command to create an IBM Storage Scale RAID scale-out recovery group (IBM Storage Scale Erasure Code Edition). This requires an mmvdisk node class containing from 4 to 32 identically equipped servers. The node class must already have been configured using the **mmvdisk server configure** command, and the servers must have been restarted with the configuration in effect. The servers must have identical IBM Storage Scale RAID disk topologies. The name of the recovery group must be supplied using the `--recovery-group` *Rgname* parameter.

Use the **mmvdisk recoverygroup create** command to create an IBM Storage Scale RAID shared recovery group (IBM Elastic Storage System 3000). This requires an mmvdisk node class containing two servers that share the same set of disks. The node class must already have been configured using the **mmvdisk server configure** command, and the servers must have been restarted with the configuration in effect. The servers must have identical IBM Storage Scale RAID disk topologies. The name of the recovery group must be supplied using the `--recovery-group` *RgName* parameter.

The **mmvdisk recoverygroup create** command treats a recovery group and its log vdisks as a unit: Creating mmvdisk recovery groups also formats the appropriate log vdisks for the recovery groups as a serial part of the process. When creating large numbers of recovery groups in sequence, it might be desired to defer the format of the log vdisks until after all the recovery groups proper have been created. This can speed the process by taking advantage of parallelization to format the deferred log vdisks all at once. The `--defer-log-format` flag can be used with **mmvdisk recoverygroup create** to postpone log vdisk formatting. After all the base recovery groups have been created, the **mmvdisk recoverygroup create --complete-log-format** command will format any deferred log vdisks. User vdisk sets can not be defined in a recovery group with deferred log vdisks until the log format is completed.

Before new mmvdisk recovery groups can be created, all existing non-mmvdisk recovery groups in a cluster must be converted to mmvdisk management.

Existing non-mmvdisk paired recovery groups can be placed under mmvdisk management with the **mmvdisk recoverygroup convert** command. This procedure is applied once per recovery group pair. The name of one or both existing recovery groups in a pair must be supplied; if only one is supplied, the other is automatically determined since they share the same two servers. A node class name must also be supplied. The node class name must not be in use, and will be created as an mmvdisk node class containing the two servers for the recovery group pair. The **mmvdisk recoverygroup convert** command examines each vdisk in the two recovery groups, and matches them automatically into converted vdisk sets. Converted vdisk sets collect from all converted recovery groups those vdisks that have identical attributes and are in the same file system. Converted vdisk sets are given constructed names such as VS002_fs2, which can later be changed to something more meaningful using the **mmvdisk vdiskset rename** command. The completion of recovery group conversion requires that the IBM Storage Scale daemon be restarted on the converted servers. This can be performed using the `--recycle` flag with the **mmvdisk recoverygroup convert** command.

Existing mmvdisk recovery group can be resized to a supported disk topology upgrade using the **mmvdisk recoverygroup resize** command. Supported upgrade paths involve adding new disk capacity in to an existing recovery group. The procedure requires careful hardware and software coordination performed by IBM Service. After the upgraded disk topology hardware has been validated, the **mmvdisk recoverygroup resize** command integrates the additional capacity into the recovery groups.

The **mmvdisk recoverygroup add** command can be used to add servers to an mmvdisk scale-out recovery group. Multiple server nodes can be added at a time, and all the servers must be configured using the **mmvdisk server configure** command. Two **mmvdisk recoverygroup add** commands are required to add a server node to a scale-out recovery group. The first **mmvdisk recoverygroup add** command adds the disk capacity of the -N Node[,Node...] servers to the recovery group and initiates a rebalance of RAID stripes across the new capacity. It creates a rebalance callback for the second command. When the rebalance has completed, the second command, **mmvdisk recoverygroup add --complete-node-add**, is invoked by the callback automatically to format the added log groups for the server and to extend any vdisk sets and file systems that use the recovery group into the new capacity.

The **mmvdisk recoverygroup replace** command can be used to replace a failed or undesired server in an mmvdisk scale-out recovery group. The undesired server is designated with the **-N** *Node* parameter, and the replacement server is designated with the `--new-node` *NewNode* parameter. The replacement server node must be equipped identically to the other servers in the recovery group (especially regarding memory and disk topology), and must first have been configured using the **mmvdisk server configure** command. Replacing a server migrates the RAID data from the old server's pdisks to the replacement server's pdisks; no changes are performed to any file systems or vdisk sets.

The **mmvdisk recoverygroup delete** command with the **-N** *Node[,Node...]* parameter can be used to delete server nodes from a scale-out recovery group. Because this removes two log groups of vdisk set members from the scale-out recovery group for each server node, all file systems that use vdisk sets from the recovery group must first be reduced by server using the **mmvdisk filesystem delete -N** *Node* command. It is not always possible to delete a server from a scale-out recovery group or file system. The reallocated disk capacity when the server's disks are removed must be able to accommodate the remaining log groups and vdisk set members, and the number of pdisks remaining in the declustered arrays must be sufficient to maintain vdisk RAID code width and VCD (vdisk configuration data) fault tolerance. The **mmvdisk filesystem delete** and **mmvdisk recoverygroup delete** commands will check first to see that these conditions will be satisfied before proceeding to delete servers from a file system or recovery group.

The **mmvdisk recoverygroup change** command can be used to change the server assignments of an mmvdisk paired recovery group, to change the recovery group feature version, or to change certain declustered array attributes. The normal use case for changing the server assignments is to swap the primary and backup servers of a paired recovery group, using the `--primary` *Node* `--backup` *Node* parameter. There is also a maintenance use case for temporarily deleting a backup server and then restoring a possibly different backup server. See *IBM Storage Scale RAID: Administration* for more information. Regardless of the use case, the recovery group will immediately switch over to being served by the new primary server. Only paired recovery groups have primary and backup server assignments.

The active server for a paired recovery group can be changed immediately, without permanently changing the primary and backup assignments, by using the **mmvdisk recoverygroup change --active** *Node* command. The new active server must be a member of the recovery group node class. For planned temporary recovery group reassignment, use `--active` for maintenance operations. If the keyword **DEFAULT** is used instead of a server name, it means to make the primary server the active server for the specified paired recovery group.

For a shared recovery group, the **mmvdisk recoverygroup change --active *Node*** command means to make the specified node the server for all four user log groups and the root log group. The specified node therefore temporarily becomes the sole active server for the entire shared recovery group, leaving the other server idle. This should only be done in unusual maintenance situations, since it is normally considered an error condition for one of the servers of a shared recovery group to be idle. If the keyword **DEFAULT** is used instead of a server name, it restores the normal default balance of log groups, making each of the two servers responsible for two user log groups.

The **mmvdisk recoverygroup change --active** command does not apply to scale-out recovery groups.

The recovery group feature version can be changed using the **mmvdisk recoverygroup change --version** {*VersionString* | LATEST} command, where the keyword LATEST means the latest feature version available with the currently installed IBM Storage Scale RAID software.

For paired and shared recovery groups, the effective pdisk spares and VCD spares (vdisk configuration data replicas) of each declustered array are set to recommended values when the recovery group is created. Under the direction of IBM Service, the effective pdisk spares can be changed using the **mmvdisk recoverygroup change --declustered-array** *DaName* command with `--spares` parameter. For paired recovery groups, the same applies to VCD spares with the `--vcd-spares` parameter. For shared recovery groups, the value of VCD spares is a function of the spare and non-spare pdisk totals and cannot be changed directly.

For scale-out recovery groups, the effective pdisk spares of each declustered array is set to a minimum allowable default value when the recovery group is created. The **mmvdisk recoverygroup change** command can be used to increase pdisk spares so that rebuild can accommodate a larger number of disk or node failures. Either the `--spare-pdisks` or `--spare-nodes` parameter can be used for the declustered arrays of scale-out recovery groups. The `--spare-pdisks` parameter gives the exact number of spare pdisks desired for the specified declustered arrays. The `--spare-nodes` parameter determines the effective spare pdisks value according to the number of pdisks each node contributes to a particular declustered array. For example, if each node in a scale-out recovery group contributes 5 pdisks to declustered array DA2, `--spare-nodes 3` will attempt to set 15 spare pdisks in DA2. The spares for scale-out recovery groups are subject to minimum and maximum values that depend on the number of nodes in the recovery group and the number of pdisks per node. As with shared recovery groups, the VCD spares setting for a scale-out recovery group declustered array is derived automatically from the spare and non-spare pdisk totals and can not be changed.

For more information on declustered array spares, see *IBM Storage Scale RAID: Administration*.

The scrub duration, replace threshold, and bit error rate of each declustered array in a recovery group are set to recommended values when the recovery group is created. They can be changed using the **mmvdisk recoverygroup change --declustered-array** *DaName* command with the `--scrub-duration` , `--replace-threshold` and `--bit-error-rate` parameters. See the *Parameters* section for more information.

The **mmvdisk recoverygroup change --suspend** command can be used to temporarily suspend a single node in a scale-out recovery group so that maintenance can be performed on the node. IBM Storage Scale is stopped on the suspended node and its pdisks are also suspended. Scale-out server maintenance should be performed quickly to avoid unnecessary RAID rebuild overhead. By default, there is a window of 20 minutes before the data on a suspended server's pdisks is rebuilt onto other nodes. The **--window** *N* option can be used to adjust the number of minutes before the suspended server's pdisks are rebuilt. The value of *N* must be a number between 10 and 60 inclusive. When node maintenance is complete, the **mmvdisk recoverygroup change --resume** command must be used to start IBM Storage Scale on the node and make its pdisks available again.

The **mmvdisk recoverygroup change --restart** command can be used to try to restart a recovery group where the servers are still active in the cluster, but one or more log groups of the recovery group are not. Restarting a recovery group is much less disruptive to a cluster than using the **mmshutdown** and **mmstartup** commands on the recovery group server node class. The normal use case for the **mmvdisk recoverygroup change --restart** command is after a recovery group has deliberately resigned due to loss of connectivity to file system vdisk NSD client nodes performing replicated writes, but it can also be used when individual log groups have resigned. If a recovery group is already completely active, the **mmvdisk recoverygroup change --restart** command will refuse to restart it.

The **mmvdisk recoverygroup change --refresh-pdisk-info** command can be used to make a recovery group reread the hardware information for all its pdisks. This can be useful if maintenance procedures have updated, for example, the pdisk firmware.

Recovery group information is listed using the **mmvdisk recoverygroup list** command. With no arguments. **mmvdisk recoverygroup list** shows all recovery groups in the cluster, mmvdisk and non-mmvdisk alike, whether they are active or not and the current server. This can be useful to tell which recovery groups have and have not been converted to mmvdisk management. With a specific `--recovery-group` *RgName* argument, this information can be displayed for a single recovery group.

The **mmvdisk recoverygroup list --declustered-array** command gives an at-a-glance summary for each declustered array in each recovery group. The **mmvdisk recoverygroup list --not-ok** command shows only those recovery groups that are down, have inactive servers, or have pdisks in need of replacement.

For a specified recovery group that is under mmvdisk management, the **mmvdisk recoverygroup list --recovery-group** *RgName* can provide more detailed information; use the following flags to choose different output sections:

- `--version` shows recovery group version information.
- `--server` shows recovery group server information.
- `--declustered-array` shows recovery group declustered array information.
- `--pdisk` shows recovery group pdisk information.
- `--log-group` shows recovery group log group information.
- `--vdisk-set` shows recovery group vdisk set information.
- `--vdisk` shows recovery group individual vdisk information.
- `--fault-tolerance` shows recovery group fault tolerance information.

Any combination of the detailed information flags can be used, or the `--all` flag can be used to show all sections.

The recovery group event log is displayed independently of the detailed information by using the **mmvdisk recoverygroup list --events** command. By default, all available events in the short- and long-term event logs are displayed. The parameters `--days` *Days*, `--short-term` *Codes*, and `--long-term` *Codes* can be combined to select a subset of the event log to display. The event *Codes* argument is a string containing one or more of the letters:

- *C* for configuration events.
- *E* for error events.
- *W* for warning events.
- *I* for informational events.
- *D* for detail events.

An mmvdisk recovery group is deleted using the **mmvdisk recoverygroup delete** command. Even though paired recovery groups are created in pairs, they are deleted individually. The order in which the two recovery groups of a pair are deleted does not matter. A recovery group can not be deleted until it is undefined from any vdisk sets. By necessity, deleting a recovery group also deletes all the log vdisks in the recovery group.

Parameters

mmvdisk recoverygroup create

Create mmvdisk recovery groups.

mmvdisk recoverygroup convert

Convert existing non-mmvdisk paired recovery groups to mmvdisk management.

mmvdisk recoverygroup resize

Resize existing mmvdisk recovery groups to a supported disk topology upgrade.

mmvdisk recoverygroup add

Add server nodes to a scale-out recovery group.

mmvdisk recoverygroup replace

Replace a server node in a scale-out recovery group.

mmvdisk recoverygroup delete

Delete an mmvdisk recovery group, or delete specified servers from a scale-out recovery group.

mmvdisk recoverygroup change

Change the characteristics of an mmvdisk recovery group.

mmvdisk recoverygroup list

List all recovery groups, or list detailed information for mmvdisk recovery groups.

--recovery-group ***Rgname1*,*RgName2***

Specifies a recovery group pair for use with the **mmvdisk recoverygroup create**, **convert**, and **resize** commands.

--recovery-group ***Rgname***

Specifies a single recovery group for use with the **mmvdisk recoverygroup create**, **add**, **replace**, **delete**, **change**, **list**, and **resize** commands.

--node-class ***NcName***

With the **mmvdisk recoverygroup create** or **mmvdisk recoverygroup convert** commands, specifies the mmvdisk node class to be associated with the recovery groups.

--match ***N***

Specifies the minimum IBM Storage Scale RAID disk topology matching metric to be accepted for recovery group creation. The default is 100 (out of 100), meaning that only perfectly matching disk topologies can be used to create recovery groups. It is strongly advised against creating recovery groups with imperfectly matching disk topologies.

--multiple-da-legacy

Specifies that multiple declustered arrays be created within recovery groups on GL4 and GL6 IBM Storage Scale RAID disk topologies. This option is provided only for compatibility with existing GL4 and GL6 installations that use multiple DAs. If compatibility with existing multiple DA installations is not required, this option should not be used.

--bit-error-rate {enable | disable}

Specifies whether bit error rate enforcement shall be enabled for this declustered array. If the bit error rate of a disk within a DA is predicted to be beyond an expected threshold, that pdisk will be marked as failing. The default setting for declustered arrays that support bit error rate enforcement is `enable`. If **mmvdisk rg list** shows a value of `N/A` under the BER column, it means that bit error rate enforcement is not supported on this declustered array and is permanently disabled.

--split-da [*DaType***:]*N*[,***DaType*:*N*...]**

With the **mmvdisk recoverygroup create** or **mmvdisk recoverygroup resize** command, specifies maximum pdisk number per node for each declustered array base on different disk types. The disk type can be `all`, `nvme`, `ssd`, or `hdd`. If specified a number without disk type, that means all declustered array use the same pdisk number per node per disk type. This option is valid for **mmvdisk recoverygroup resize** command, when the recovery group was created without the `--split-da` option. This option is supported on the scale-out recovery group only.

--trim-da ***DaType*[,*DaType*...]**

With the **mmvdisk recoverygroup create** command, specify that the trim function should be enabled for declustered arrays of the specified disk hardware types. The disk hardware type can be `all` or a comma-separated list chosen from `nvme`, `ssd`, or `hdd`. The default is to not enable the trim function for any declustered array in a recovery group. The IBM Storage Scale RAID daemon will refuse to create a recovery group when trim is requested on a disk hardware type that is not supported. See *IBM Storage Scale RAID: Administration* for a description of the declustered array trim function.

--trim-da {yes | no}

With the **mmvdisk recoverygroup change** command, specify that the trim function should be enabled or disabled for a declustered array. The IBM Storage Scale RAID daemon will refuse to enable trim on a declustered array with a disk hardware type that is not supported. See *IBM Storage Scale RAID: Administration* for a description of the declustered array trim function.

--log-home-da *DaName*

With the **mmvdisk recoverygroup create** command for scale-out recovery groups with multiple declustered arrays, choose the declustered array where log home vdisks are to be placed. The default is to choose the SSD/NVMe DA with the greatest number of pdisks; or, if no SSD/NVMe DA is present, the HDD DA with the greatest number of pdisks.

--disk-list *DiskExpr*
--ignore-disk-list *DiskExpr*

With the **mmvdisk recoverygroup create, add, replace or resize** commands, the `--disk-list` option specifies the disk list to be used and the `--ignore-disk-list` option ignores the disk list for recovery group. The `--ignore-disk-list` option has higher priority than the `--disk-list` option. That is, if some disks are specified in `--disk-list` and `--ignore-disk-list`, it will be ignored to create a recovery group.

DiskExpr is the expression of a group of disks or enclosure location id. It supports simple way to represent a group of disks or enclosure location ids. It uses '[]' to represent a set of digits or letters, '-' to simplify continuous digit or letter, ',' to separate different set and ';' to separate different node disk expression. [1-5] means 1,2,3,4,5. nvme[0-9]n1 means nvme0n1,nvme1n1,nvme2n1,nvme3n1, ... nvme9n1 10 disks.

Example:

1-9:

Location 1 through 9 on each node. In most situation, the location format is {enclosure id}-{location id}. If each node only includes one enclosure id, the enclosure id could be ignored. In fact, for ECE, the enclosure id is the host's 'Serial Number'. It is unique for each node. For ESS3200, each node has only one enclosure. So, the enclosure id could be ignored.

1-9;Node1:2-10

Location 1 through 9 on each node except Node1. Location 2 through 10 on Node1

sd[b-d];Node1:sd[b-c,e]

sdb,sdc,sdd on each node except Node1. sdb,sdc,sde on Node1.

nvme[0-4]n1

nvme0n1,nvme1n1,nvme2n1,nvme3n1,nvme4n1 on each node.

nvme[0-4]n1;Node1:nvme1n[1-5]

nvme0n1,nvme1n1,nvme2n1,nvme3n1,nvme4n1 on each node except Node1.
nvme1n1,nvme1n2,nvme1n3,nvme1n4,nvme1n5 on Node 1.

DiskExpr could be a file whose content is disk list also. The file content could be obtained by following command:

```
mmvdisk server list --nc {NcName} --disk-list -Y | mmyfields -F: nodeName diskListRaw > /tmp/
diskExpr.out
```

You can edit this file to reserve the disks that are used for a recovery group, then specify the file name as the value of `--disk-list` to create a recovery group. The following command can be used to verify whether `diskExpr` is proper:

```
mmvdisk server list --nc {NcName} --disk-list diskExpr --disk-topology
```

With the **mmvdisk recoverygroup add** and **replace** commands, `--disk-list` and `--ignore-disk-list` are applied on the newly added nodes only.

With the **mmvdisk recoverygroup resize** command, `--disk-list` and `--ignore-disk-list` are applied on the newly added disks only.

This option is not supported on a paired recovery group.

-v {yes | no}

With the **mmvdisk recoverygroup create**, **resize**, **add**, and **replace** commands, specifies whether pdisks should be verified to be empty of IBM Storage Scale RAID data before they are initialized in a recovery group. The default is *yes*, verify that pdisks are empty of data. With the **mmvdisk recoverygroup change** `--primary` *Node* [`--backup` *Node*] command, specifies whether or not to verify that the recovery group pdisks are present on the new servers. The default is *yes*.

--fanout *N*

Specifies how many servers to query simultaneously for their disk topologies. The default is 32.

--defer-log-format

Specifies that the log vdisks should not be formatted at the time recovery groups are created.

--complete-log-format

Specifies that all deferred log vdisks in any recovery groups should be formatted.

-N *Node*

With the **mmvdisk recoverygroup replace** command, specifies the server node to be replaced.

-N *Node[,Node...]*

With the **mmvdisk recoverygroup add** and **delete** commands, specifies the server nodes to be added or deleted.

--new-node *NewNode*

With the **mmvdisk recoverygroup replace** command, specifies the replacement server node to take the place of the node being replaced.

--complete-node-add

With the **mmvdisk recoverygroup add** command, complete the addition of a scale-out recovery group server by creating new log groups and extending the vdisk sets and file systems that use the recovery group. This action will be invoked by relanace callback automatically once the rebalance has completed. It will verify that the new server's pdisks have rebalanced to make sufficient fault tolerance available for the new log groups and vdisks.

--inherit-config {yes | no}

With the **mmvdisk recoverygroup convert** command, specifies whether the configuration settings of non-mmvdisk managed recovery groups are preserved after conversion. The default value is *yes*, which means the configuration settings are preserved. If *no* is specified, **mmvdisk** command uses the recommended settings to overwrite the configuration of the non-mmvdisk recovery groups.

--pagepool {*n*M | *n*G | *n*%}

With the **mmvdisk recoverygroup convert** command and `--inherit-config` specified as *no*, choose a specific IBM Storage Scale pagepool value. Normally, **mmvdisk recoverygroup convert** chooses the best appropriate pagepool size for the real memory available in the server node class, so this option should only be used under instruction from IBM.

--recycle {none | one | all | *Number*}

With the **mmvdisk recoverygroup convert** command and `--inherit-config` specified as *no*, specifies the number of nodes to be simultaneously restarted so that the converted IBM Storage Scale configuration changes can take effect. The default is `none`, meaning that the administrator is responsible for using the **mmshutdown** and **mmstartup** commands to recycle the nodes or node class. The keyword `one` will recycle one node at a time, the keyword `all` will

recycle all specified nodes simultaneously, or a *Number* of nodes can be chosen to be recycled at the same time. Care should be taken when recycling nodes if it is desired to maintain quorum availability in the IBM Storage Scale cluster.

--dry-run

With the **mmvdisk recoverygroup convert** command, specifies that the process of converting an existing non-mmvdisk recovery group pair merely be simulated, without making any actual changes to the cluster configuration. This will show the conversion and vdisk set discovery process and allow it to be examined prior to making the changes permanent. With the **mmvdisk recoverygroup resize** command, `--dry-run` can be used to validate that a recovery group pair is able to be resized to a supported disk topology upgrade, without actually performing the resize.

--sanitize

Erases data securely from the drives of the deleted recovery group.

--primary ***Node*** **[--backup** ***Node*****]**

Specifies the new primary and backup servers for a recovery group. Caution must be used when omitting a backup server, as this reduces recovery group availability. See *IBM Storage Scale RAID: Administration* for more information.

--active {*Node* | DEFAULT}

With the **mmvdisk recoverygroup change** command, manipulate the current active servers for paired and shared recovery groups. This parameter does not apply to scale-out recovery groups. For a paired recovery group, specifying the node name of one of the two servers makes the specified node the current active server for the recovery group. Specifying the keyword **DEFAULT** makes the normal primary server the current active server. For a shared recovery group, specifying the node name of one of the two servers makes the specified node the active server for all log groups of the recovery group, and leaves the other server idle. Specifying the keyword **DEFAULT** restores the normal balance where each of the two servers serves two of the four user log groups.

--version {*VersionString* | Latest}

With the **mmvdisk recoverygroup change** command, specifies that the recovery group feature version be updated. The keyword LATEST updates to the latest version available with the currently installed IBM Storage Scale RAID software.

--declustered-array ***DaName***

With the **mmvdisk recoverygroup change** command, specifies a target declustered array for which attributes are to be changed.

--spares ***NumberOfSpares***

For paired or shared recovery groups with the **mmvdisk recoverygroup change** command, specifies the number of effective spares to be reserved in the target declustered array. The default depends on the recovery group IBM Storage Scale RAID disk topology, and should only be changed under the guidance of IBM Service. For scale-out recovery groups, the **--spare-pdisks** or **--spare-nodes** option must be used.

--vcd-spares ***NumberOfVcdSpares***

For paired recovery groups with the **mmvdisk recoverygroup change** command, specifies the number of disks that can become unavailable while still maintaining replication of vdisk configuration data (VCD). The default depends on the recovery group IBM Storage Scale RAID disk topology, and should only be changed under the guidance of IBM Service. This option is not valid for shared or scale-out recovery groups.

--scrub-duration ***NumberOfDays***

With the **mmvdisk recoverygroup change** command, specifies the background rate at which vdisks should be checked for errors. The value is the number of days (from 1 to 365) that the process should take for the entire target declustered array. The default is 14.

--replace-threshold *NumberOfPdisks*

With the **mmvdisk recoverygroup change** command, specifies the number of out-of-service pdisks in the target declustered array at which point the recovery group will report that it needs service for disk replacement. The default is 2 for DAs with more than 12 pdisks, or 1 for DAs with 12 or fewer.

--declustered-array {all | *DaName*[,*DaName*...]}

For a scale-out recovery group with the **mmvdisk recoverygroup change** command, specifies the target declustered arrays for the `--spare-pdisks` or `--spare-nodes` parameter.

--spare-pdisks {DEFAULT | *N*}

For a scale-out recovery group with the **mmvdisk recoverygroup change** command, specifies the number of effective spares to be reserved in the target declustered arrays. The default depends on the number of nodes in the recovery group and the number of pdisks in the declustered array. The keyword **DEFAULT** can be used to reset the effective spares to the default for the target declustered arrays.

--spare-nodes {DEFAULT | *N*}

For a scale-out recovery group with the **mmvdisk recoverygroup change** command, specifies the number of effective spares in terms of the number of pdisks each node contributes to the target declustered arrays. For example, if each node contributes 3 pdisks to DA1 and 7 pdisks to DA2, **mmvdisk recoverygroup change --declustered-array DA1,DA2 --spare-nodes 2** will set 6 effective spares in DA1 and 14 effective spares in DA2, permitting those declustered arrays to completely rebuild from 2 failed nodes. The keyword **DEFAULT** can be used to reset the effective spares to the default for the target declustered arrays.

--suspend -N *Node*

For a scale-out recovery group with the **mmvdisk recoverygroup change** command, specifies a node to be suspended for maintenance. IBM Storage Scale is stopped on the node and its pdisks are suspended. Only one node can be suspended at a time.

--window *N*

With the `--suspend` option, specify the number of minutes before the data on a suspended server's pdisks is rebuilt onto other nodes. The value of *N* must be a number between 10 and 60 inclusive. The default is 20 minutes if not specified.

--resume -N *Node*

For a scale-out recovery group with the **mmvdisk recoverygroup change** command, specifies a node to be resumed. IBM Storage Scale is started on the node and its pdisks are resumed.

--restart

With the **mmvdisk recoverygroup change** command, restart a recovery group or any inactive log groups within a recovery group. This is for the special case where the recovery group servers are still active but the recovery group itself or log groups within the recovery group have been resigned. If the recovery group and all of its log groups are already active, this command has no effect. If the servers for the recovery group are not active, this command will have no effect; to start a recovery group where the servers are down, use the **mmstartup -N** ***nodeClass*** command, where ***nodeClass*** is the mmvdisk server node class for the recovery group.

--refresh-pdisk-info

With the **mmvdisk recoverygroup change** command, instructs the recovery group to query each pdisk and reread its hardware information.

--version

Specifies that recovery group version information should be listed.

--server

Specifies that recovery group server information should be listed.

--declustered-array

Specifies that recovery group declustered array information should be listed.

--pdisk

Specifies that recovery group pdisk information should be listed.

--log-group

Specifies that recovery group log group information should be listed.

--vdisk-set

Specifies that recovery group vdisk set information should be listed.

--vdisk

Specifies that recovery group vdisk information should be listed.

--fault-tolerance

Specifies that recovery group fault tolerance information should be listed.

--all

Specifies that all sections of recovery group information should be listed.

--not-ok

With the **mmvdisk recoverygroup list** command, shows those recovery groups that are down, have inactive servers, or have pdisks in need of replacement.

--events

With the **mmvdisk recoverygroup list** command, specifies that the recovery group event log should be listed.

--days *Days*

Specifies the number of past days for which recovery group events are to be displayed. The default is to display events as far back as recovery group creation.

--decimal

Specifies that recovery group event timestamps be displayed in decimal format.

--long-term *Codes*

Specifies that long-term recovery group events that correspond to the *Codes* string are displayed.

--short-term *Codes*

Specifies that short-term recovery group events that correspond to the *Codes* string are displayed.

--confirm

Confirms the deletion of a recovery group. This flag bypasses the interactive prompt that would otherwise be printed to confirm recovery group deletion.

-p

With the **mmvdisk recoverygroup delete** command, the -p flag indicates that the recovery group is "permanently damaged" and must be deleted without accessing its pdisks. This option is not permitted if the recovery group is active. This is an unusual maintenance case that should only be performed under the direction of IBM Service. Before a recovery group can be deleted using -p, any user vdisks and vdisk sets must be deleted from the recovery group using the **mmvdisk vdiskset delete -p** command for each vdisk set, followed by undefining the recovery group from each vdisk set using the **mmvdisk vdiskset undefine** command (note that -p is not necessary to undefine a damaged recovery group from a vdisk set, since the actual member vdisks in the recovery group have been deleted).

--sanitize
Erases data securely from the drives of the deleted recovery group.

-Y

Specifies that the **mmvdisk recoverygroup list** command produce colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk recoverygroup** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

- To create paired recovery groups ESS01L and ESS01R using the mmvdisk node class ESS01:

```
mmvdisk recoverygroup create --node-class ESS01 --recovery-group ESS01L,ESS01R
```

The system displays output similar to the following:

```
mmvdisk: Checking node class configuration.
mmvdisk: Checking daemon status on node 'ess01io1.gpfs.net'.
mmvdisk: Checking daemon status on node 'ess01io2.gpfs.net'.
mmvdisk: Analyzing disk topology for node 'ess01io1.gpfs.net'.
mmvdisk: Analyzing disk topology for node 'ess01io2.gpfs.net'.
mmvdisk: Creating recovery group 'ESS01L'.
mmvdisk: Creating recovery group 'ESS01R'.
mmvdisk: Formatting log vdisks for recovery groups.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIPBACKUP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIPBACKUP
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGHOME
mmvdisk: Created recovery groups 'ESS01L' and 'ESS01R'.
```

- To list all recovery groups present in the cluster:

```
mmvdisk recoverygroup list
```

The system displays output similar to the following:

```
                                                          needs    user
recovery group  active   current or master server        service  vdisks  remarks
--------------  -------  -------------------------------  -------  ------  -------
ESS01L          yes      ess01io1.gpfs.net                no            2
ESS01R          yes      ess01io2.gpfs.net                no            2
```

- To list the declustered array information for recovery group BB01L:

```
mmvdisk recoverygroup list --recovery-group BB01L --declustered-array
```

The system displays output similar to the following:

```
                declustered                   current      allowable
recovery group    arrays     vdisks  pdisks  format version format version
----------------  ---------  ------  ------  -------------- --------------
BB01L                    1       7      49  5.1.2.0        5.1.2.0
```

```
declustered    needs                          replace                          scrub       background activity
   array     service  vdisks  pdisks  spares  threshold  BER    trim  free space  duration  task   progress  priority
 ----------- -------  ------  ------  ------  ---------  ------ ----  ----------  --------  -------------------------
 DA1         yes           7      49    2,27          2  enable no       338 TiB    4 days  scrub       74%  low
```

- To convert existing recovery groups ESS01L and ESS01R with a new mmvdisk node class ESS01:

```
mmvdisk recoverygroup convert --recovery-group ESS01L,ESS01R --node-class ESS01 --inherit-
config no
```

The system displays output similar to the following:

```
mmvdisk: This command will permanently change the GNR configuration
mmvdisk: attributes and disable the legacy GNR command set for the
mmvdisk: servers and recovery groups involved, and their subsequent
mmvdisk: administration must be performed with the mmvdisk command.

mmvdisk: Do you wish to continue (yes or no)? yes

mmvdisk: Converting recovery groups 'ESS01L' and 'ESS01R'.
mmvdisk: Creating node class 'ESS01'.
mmvdisk: Adding 'ess01io1' to node class 'ESS01'.
mmvdisk: Adding 'ess01io2' to node class 'ESS01'.
mmvdisk: Associating recovery group 'ESS01L' with node class 'ESS01'.
mmvdisk: Associating recovery group 'ESS01R' with node class 'ESS01'.
mmvdisk: Recording pre-conversion cluster configuration in /var/mmfs/tmp/
mmvdisk.convert.ESS01L.ESS01R.before.m24
mmvdisk: Updating server configuration attributes.
mmvdisk: Checking resources for specified nodes.
mmvdisk: Setting configuration for node class 'ESS01'.
mmvdisk: Defining vdisk set 'VS001_fs3' with recovery group 'ESS01L' (vdisk 'ESS01Lv3fs3').
mmvdisk: Defining vdisk set 'VS002_fs4' with recovery group 'ESS01L' (vdisk 'ESS01Lv4fs4').
mmvdisk: Defining vdisk set 'VS003_fs1' with recovery group 'ESS01L' (vdisk 'ESS01Lv1fs1data').
mmvdisk: Defining vdisk set 'VS004_fs2' with recovery group 'ESS01L' (vdisk 'ESS01Lv2fs2').
mmvdisk: Defining vdisk set 'VS005_fs1' with recovery group 'ESS01L' (vdisk 'ESS01Lv1fs1meta').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS003_fs1' (vdisk 'ESS01Rv1fs1data').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS004_fs2' (vdisk 'ESS01Rv2fs2').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS001_fs3' (vdisk 'ESS01Rv3fs3').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS005_fs1' (vdisk 'ESS01Rv1fs1meta').
mmvdisk: Adding recovery group 'ESS01R' to vdisk set 'VS002_fs4' (vdisk 'ESS01Rv4fs4').
mmvdisk: Committing cluster configuration changes.
mmvdisk: Recording post-conversion cluster configuration in /var/mmfs/tmp/
mmvdisk.convert.ESS01L.ESS01R.after.m24

mmvdisk: For configuration changes to take effect, GPFS should be restarted
mmvdisk: on node class 'ESS01'.
```

- To create a scale-out recovery group RG01 using the mmvdisk node class NC01 that contains servers s01, s02, s03, s04, and s05:

```
mmvdisk recoverygroup create --node-class NC01 --recovery-group RG01
```

The system displays output similar to the following:

```
mmvdisk: Checking node class configuration.
mmvdisk: Checking daemon status on node 's04'.
mmvdisk: Checking daemon status on node 's02'.
mmvdisk: Checking daemon status on node 's01'.
mmvdisk: Checking daemon status on node 's05'.
mmvdisk: Checking daemon status on node 's03'.
mmvdisk: Analyzing disk topology for node 's04'.
mmvdisk: Analyzing disk topology for node 's02'.
mmvdisk: Analyzing disk topology for node 's01'.
mmvdisk: Analyzing disk topology for node 's05'.
mmvdisk: Analyzing disk topology for node 's03'.
mmvdisk: Creating recovery group 'RG01'.
mmvdisk: Formatting log vdisks for recovery group.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001ROOTLOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG001LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG002LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG003LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG004LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG005LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG006LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG007LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG008LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG009LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG010LOGHOME
mmvdisk: Created recovery group 'RG01'.
```

- To perform the first step in adding server node s06 to scale-out recovery group RG01:

```
mmvdisk recoverygroup add --recovery-group RG01 -N s06
```

The system displays output similar to the following:

```
mmvdisk: Checking daemon status on node 's06'.
mmvdisk: Checking resources for specified nodes.
mmvdisk: Adding 's06' to node class 'NC01'.
mmvdisk: Analyzing disk topology for node 's04'.
mmvdisk: Analyzing disk topology for node 's02'.
mmvdisk: Analyzing disk topology for node 's01'.
mmvdisk: Analyzing disk topology for node 's05'.
mmvdisk: Analyzing disk topology for node 's03'.
mmvdisk: Analyzing disk topology for node 's06'.
mmvdisk: Updating server list for recovery group 'RG01'.
mmvdisk: Updating pdisk list for recovery group 'RG01'.
mmvdisk: The following pdisks will be formatted on node s04:
mmvdisk:     //s06/dev/sdg
mmvdisk:     //s06/dev/sdh
mmvdisk:     //s06/dev/sdf
mmvdisk:     //s06/dev/sde
mmvdisk:     //s06/dev/sdc
mmvdisk:     //s06/dev/sdd
mmvdisk:     //s06/dev/sdb
mmvdisk: Updating parameters for declustered array 'DA1'.
mmvdisk: Updating parameters for declustered array 'DA2'.
mmvdisk: Node 's06' added to recovery group 'RG01'.
mmvdisk: Log group and vdisk set operations for recovery group 'RG01'
mmvdisk: will do until rebalance completes in all declustered arrays.
mmvdisk: A callback 'RG001CompletNodeAdd' has been created to monitor the rebalance state.
mmvdisk: Once rebalance completes, log group and vdisk set will be created automatically.
```

- To complete the addition of node s06 to scale-out recovery group RG01:

```
mmvdisk recoverygroup add --recovery-group RG01 --complete-node-add
```

The system displays output similar to the following:

```
mmvdisk: Verifying that the DAs in recovery group 'RG01' are idle.
mmvdisk: Updating log vdisks for recovery group 'RG01'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG011LOGHOME
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG012LOGHOME
mmvdisk: Updating vdisk sets for recovery group 'RG01.'
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'vs01'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG011VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LG012VS001
mmvdisk: Created all vdisks in vdisk set 'vs01'.
mmvdisk: (mmcrnsd) Processing disk RG001LG011VS001
mmvdisk: (mmcrnsd) Processing disk RG001LG012VS001
mmvdisk: Created all NSDs in vdisk set 'vs01'.
mmvdisk: Extending file system 'fs01'.
mmvdisk: The following disks of fs01 will be formatted on node s01:
mmvdisk:     RG001LG011VS001: size 1365 GB
mmvdisk:     RG001LG012VS001: size 1365 GB
mmvdisk: Extending Allocation Map
mmvdisk: Checking Allocation Map for storage pool system
mmvdisk: Completed adding disks to file system fs01.
```

- To replace server s06 of recovery group RG01 with new server s07:

```
mmvdisk recoverygroup replace --recovery-group RG01 -N s06 --new-node s07
```

The system displays output similar to the following:

```
mmvdisk: Checking daemon status on node 's07'.
mmvdisk: Analyzing disk topology for node 's04'.
mmvdisk: Analyzing disk topology for node 's02'.
mmvdisk: Analyzing disk topology for node 's01'.
mmvdisk: Analyzing disk topology for node 's05'.
mmvdisk: Analyzing disk topology for node 's03'.
mmvdisk: Analyzing disk topology for node 's07'.
mmvdisk: Adding 's07' to node class 'NC01'.
mmvdisk: Checking resources for specified nodes.
mmvdisk: Updating server list for recovery group 'RG01'.
mmvdisk: Updating pdisk list for recovery group 'RG01'.
mmvdisk: This could take a long time.
mmvdisk: Adding node 's07' pdisks to recovery group 'RG01'.
mmvdisk: The following pdisks will be formatted on node s04:
mmvdisk:     //s07/dev/sdc
mmvdisk:     //s07/dev/sde
mmvdisk:     //s07/dev/sdg
mmvdisk:     //s07/dev/sdb
mmvdisk:     //s07/dev/sdd
mmvdisk:     //s07/dev/sdf
```

```
mmvdisk:     //s07/dev/sdh
mmvdisk: Deleting node 's06' pdisks from recovery group 'RG01'.
mmvdisk: Removing node 's06' from node class 'NC01'.
mmvdisk: Updating server list for recovery group 'RG01'.
```

- To list all declustered arrays in all recovery groups in a concise format:

```
mmvdisk recoverygroup list --declustered-array
```

The system displays output similar to the following:

```
                declustered   needs                                          capacity             pdisks
recovery group     array     service  type   BER     trim   total raw free raw free%  total spare  background task
--------------  -----------  -------  ----  -----    ----   --------- -------- -----  ----- -----  ---------------
BB01L           DA1          no       HDD   enable   no     421 TiB   336 TiB   79%     49     2   scrub (84%)
BB01R           DA1          no       HDD   enable   no     448 TiB   358 TiB   80%     52     2   scrub (49%)
```

- To move all log groups in the shared recovery group ESS3000 to the server `canister01`, leaving the server `canister02` idle:

```
mmvdisk recoverygroup change --recovery-group ESS3000 --active canister01
```

The system displays output similar to the following:

```
mmvdisk: Waiting up to 5 minutes for log groups to change servers.

 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  canister01                      yes      serving ESS3000: root, LG001, LG002, LG003, LG004
     2  canister02                      yes      configured
```

- To see how a shared recovery group running on only one of the two servers is reported as an error condition:

```
mmvdisk recoverygroup list --not-ok
```

The system displays output similar to the following:

```
recovery group  remarks
--------------  -------
ESS3000         server canister01 has 4 user log groups
                server canister02 has 0 user log groups
```

- To restart recovery group ECERG1:

```
mmvdisk recoverygroup change --recovery-group ECERG1 --restart
```

The system displays output similar to the following:

```
mmvdisk: Waiting up to 5 minutes for recovery group 'ECERG1' to restart.

 node
number  server                          active   remarks
------  ------------------------------  -------  -------
     1  server1                         yes      serving ECERG1: LG003, LG006
     2  server2                         yes      serving ECERG1: LG002, LG008
     3  server3                         yes      serving ECERG1: LG005, LG011
     4  server4                         yes      serving ECERG1: LG001, LG007
     5  server5                         yes      serving ECERG1: LG004, LG010
     6  server6                         yes      serving ECERG1: root, LG009, LG012

mmvdisk: Recovery group 'ECERG1' has been restarted.
```

See also

- “mmvdisk command” on page 341
- “mmvdisk nodeclass command” on page 346
- “mmvdisk server command” on page 350
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380

- “mmvdisk vdisk command” on page 386
- “mmvdisk pdisk command” on page 388

Location
/usr/lpp/mmfs/bin

mmvdisk filesystem command

Manages mmvdisk file systems for IBM Storage Scale RAID.

Synopsis

```
mmvdisk filesystem create  --file-system FsName --vdisk-set VdiskSet[,VdiskSet...]
                           [--failure-groups NodeClass=FG[,NodeClass=FG...]]
                           [--trim {auto | no}] [--mmcrfs mmcrfs-options]
```

or

```
mmvdisk filesystem add  --file-system FsName --vdisk-set VdiskSet[,VdiskSet...]
                        [--failure-groups NodeClass=FG[,NodeClass=FG...]]
                        [--trim {auto | no}]
```

or

```
mmvdisk filesystem delete  --file-system FsName --vdisk-set VdiskSet[,VdiskSet...]
                           [--recovery-group RgName [-N Node]] [--confirm]
```

or

```
mmvdisk filesystem delete  --file-system FsName [--confirm]
```

or

```
mmvdisk filesystem change  --file-system FsName
                           {--register |
                           --trim {auto | no} [--vdisk-set Name[,Name...]]|
                           --failure-groups [NodeClass=FG[,NodeClass=FG...]]}
                           [--confirm]
```

or

```
mmvdisk filesystem list  [--file-system FsName] [-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk filesystem** command to create mmvdisk file systems using IBM Storage Scale RAID vdisk sets. The **mmvdisk filesystem** command can also be used to add vdisk sets to an mmvdisk or IBM Storage Scale file system; to delete vdisk sets from an mmvdisk file system; to delete an mmvdisk file system in its entirety; to change an mmvdisk file system; and to list mmvdisk file systems and show the vdisk sets they contain.

An mmvdisk file system is an IBM Storage Scale file system where each vdisk NSD in the file system is from a vdisk set. The **mmvdisk filesystem** command applies IBM Storage Scale RAID best practices to the management of file systems constructed from vdisk sets. One important such best practice is that the

member vdisk NSDs of a given vdisk set can belong only to one file system, but a file system can contain multiple vdisk sets.

The **mmvdisk filesystem create** command takes a file system device name and one or more vdisk sets, and creates an IBM Storage Scale file system from the member vdisk NSDs of the vdisk sets. The file system device name can not already be in use, and the member vdisk NSDs of the vdisk sets must be created. File system NSD usage, storage pools, and block sizes are set according to the vdisk set attributes. File systems built from vdisk NSDs always use the `-j scatter` option of the **mmcrfs** command. File system failure groups for the vdisk NSDs are automatically assigned appropriate values, depending on the data and metadata replication values and the node classes and recovery groups in the vdisk sets. If the vdisk sets and maximum replication values meet certain conditions, custom failure group values can be assigned on a per-node class basis using the `--failure-groups` option.

Additional IBM Storage Scale file system options supported by the **mmcrfs** command (for example, quota enablement or a custom mount point) can be specified using the `--mmcrfs` flag: Any parameters on the command line that appear after the `--mmcrfs` flag are not interpreted by **mmvdisk**, but are instead passed unchanged to the **mmcrfs** command. However, the **mmvdisk filesystem create** command will inspect these parameters to ensure that `-j` cluster is not specified, and the `-m` and `-r` default replication settings and the `-M` and `-R` maximum replication settings will, if present, be used to guide the assignment of automatic failure group values. If replication settings are not specified when creating a file system, the `-m` and `-r` default replication settings are both 1 and the `-M` and `-R` maximum replication settings are both 2. Note that the IBM Storage Scale maximum replication settings cannot be changed after a file system is created.

If the file system metadata and data maximum replication settings are both equal to 2 or are both equal to 3, it is possible to use the `--failure-groups` option to assign custom failure group values to the file system vdisk NSDs on a per-node class basis. The node class serves as an identifier for the single point of failure (a failure group) of the vdisk NSDs in the associated scale-out recovery group or in the two paired recovery groups of the node class. To assign custom failure group values, the number of node classes in use across the file system must be equal to or greater than the maximum replication setting, and the number of distinct failure group values assigned to the node classes must also be equal to or greater than the maximum replication setting. The failure group FG values must be a whole number from 1 to 65535.

Use the `--trim auto` option with the **mmvdisk filesystem create** command to enable vdisk NSD trim for capable file system vdisk set member NSDs. The default value is consistent with the corresponding declustered array for each vdisk NSD. See *IBM Storage Scale RAID: Administration* for a description of the vdisk NSD trim function.

The **mmvdisk filesystem add** command is an interface between vdisk sets, mmvdisk file systems, and the IBM Storage Scale **mmadddisk** command. Use the **mmvdisk filesystem add** command to add new or newly extended vdisk sets to an existing mmvdisk file system. The member vdisk NSDs of the vdisk sets must be created using **mmvdisk vdiskset create** before they can be added to a file system. When a vdisk set has been extended by adding recovery groups, the **mmvdisk filesystem add** command only adds those member vdisk NSDs that are not already in the file system. Custom failure group assignments can only be made for vdisk NSDs from node classes that are not already represented in the file system; if the node class for a new vdisk NSD is already present in the file system, new vdisk NSDs from that node class must use the existing failure group value. To change the failure group values of existing node classes, the **mmvdisk filesystem change** command can be used after vdisk sets are added.

The **mmvdisk filesystem add** command can also add vdisk sets to an existing non-vdisk IBM Storage Scale file system. In this case, the file system becomes an **mmvdisk** file system with both non-vdisk NSDs and vdisk set NSDs. Custom failure group assignments are permitted if the same conditions are met as required by **mmvdisk filesystem create**.

To add non-vdisk NSDs to an mmvdisk file system, the IBM Storage Scale **mmadddisk** command must be used.

Use the `--trim auto` option with the **mmvdisk filesystem add** command to enable vdisk NSD trim for capable file system vdisk NSDs that are being added. If the file system already contains members of a specified vdisk set that have vdisk NSD trim enabled, new members from the same vdisk set will automatically have vdisk NSD trim enabled. Otherwise, the default value is consistent with the

corresponding declustered array for each vdisk NSD. See *IBM Storage Scale RAID: Administration* for a description of the vdisk NSD trim function.

Use the **mmvdisk filesystem delete** command to delete vdisk sets or recovery groups or an individual scale-out recovery group node from an mmvdisk file system, or to delete an mmvdisk file system in its entirety. When deleting vdisk sets or entire recovery groups from an mmvdisk file system, the **mmvdisk filesystem delete** command is an interface between vdisk sets, mmvdisk file systems, and the IBM Storage Scale **mmdeldisk** command is specified without the `--confirm` option. When deleting an mmvdisk file system in its entirety, the **mmvdisk filesystem delete** command is an interface between vdisk sets, mmvdisk file systems, and the IBM Storage Scale **mmdelfs** command is specified without the `--confirm` option.

Deleting an individual scale-out recovery group server node from an mmvdisk file system should only be performed as a prelude to deleting the entire node from its scale-out recovery group. For each specified vdisk set with members in the specified recovery group, **mmdeldisk** is used to remove the vdisk NSDs from the two highest numbered log groups of the recovery group (note that vdisk NSDs are not specific to server nodes, but instead belong to log groups of the recovery group, and therefore the specified node does need to be active to have "its" vdisk NSDs deleted). Until the node is itself entirely deleted from the recovery group, these vdisk NSDs will be orphaned from the file system that their vdisk set belongs to.

It is not always possible to delete a node from a scale-out recovery group or file system. The reallocated disk capacity when the server node's disks are removed must be able to accommodate the remaining log groups and vdisk set members, and the number of pdisks remaining in the declustered arrays must be sufficient to maintain vdisk RAID code width and VCD (vdisk configuration data) fault tolerance. The **mmvdisk filesystem delete -N** *Node* command will check first to see that these conditions will be satisfied before proceeding to delete a server node from a file system.

Deleting vdisk sets from a file system will always ask for confirmation, since the underlying **mmdelvdisk** command can take a very long time to migrate data off the vdisk NSDs that are being deleted. This prompt cannot be bypassed.

Deleting an entire mmvdisk file system will always ask for confirmation, since this will destroy all file system data. This prompt can be bypassed and answered in the affirmative with the `--confirm` flag.

Use the **mmvdisk filesystem change** command to reassign the failure group values for an mmvdisk file system. To have mmvdisk assign the failure group values automatically, use the `--failure-groups` flag without any arguments. Custom failure group assignments are permitted on a per-node class basis using the `--failure-groups` *NodeClass=FG*[,*NodeClass=FG*...] parameter, provided that the same conditions are met as required by mmvdisk filesystem create. In either case, changing the failure group assignments for a file system should be followed by **mmrestripefs -r** to ensure proper replication using the new failure group assignments.

Use the `--trim {auto | no}` option with the **mmvdisk filesystem change** command to change whether vdisk NSD trim is enabled or disabled for capable file system vdisk set member NSDs. See *IBM Storage Scale RAID: Administration* for a description of the vdisk NSD trim function.

Use the `--register` option with the **mmvdisk filesystem change** command to make sure that the association between a file system and its vdisk sets is properly registered by the **mmvdisk** command. This is useful for certain unusual mixed vdisk and non-vdisk file systems, or if somehow the association between a file system and its vdisk sets is not registered.

Use the **mmvdisk filesystem list** command to list mmvdisk file systems and the file system NSD information for their vdisk sets.

The **mmvdisk filesystem** command manages only the vdisk NSD aspects of an mmvdisk file system. Other IBM Storage Scale file system attributes continue to be managed by other IBM Storage Scale file system commands; for example, the **mmlsfs** and **mmchfs** commands.

Parameters

mmvdisk filesystem create
: Create mmvdisk file systems using IBM Storage Scale RAID vdisk sets.

mmvdisk filesystem add
Add IBM Storage Scale RAID vdisk sets to mmvdisk file systems.

mmvdisk filesystem delete
Delete IBM Storage Scale RAID vdisk sets from mmvdisk file systems, or delete an mmvdisk file system in its entirety.

mmvdisk filesystem change
Change the failure group assignments of an mmvdisk file system.

mmvdisk filesystem list
List mmvdisk file system or their component IBM Storage Scale RAID vdisk sets.

--file-system ***FsName***
Specifies the name of the file system to be operated upon.

--vdisk-set ***VdiskSet*[,*VdiskSet*...]**
Specifies the vdisk sets to be affected by a file system operation.

--mmcrfs

With the **mmvdisk filesystem create** command, the `--mmcrfs` flag indicates that all following command line parameters are not to be interpreted by **mmvdisk**, but are instead to be passed to the IBM Storage Scale **mmcrfs** command.

--recovery-group ***RgName***
With the **mmvdisk filesystem delete** command, restricts vdisk set deletion to specified recovery group. This is required preparation for removing a recovery group from a vdisk set definition.

-N ***Node***
With the **mmvdisk filesystem delete --recovery-group RgName** command, restricts vdisk set deletion to a single node of the specified scale-out recovery group. This is required preparation for removing the server node from a scale-out recovery group.

--failure-groups
With the **mmvdisk filesystem change** command, performs an automatic reassignment of the failure groups for a file system's vdisk NSDs.

--failure-groups ***NodeClass=FG*[,*NodeClass=FG*...]**
With the **mmvdisk filesystem create**, **mmvdisk filesystem add**, and **mmvdisk filesystem change** commands, assigns custom failure group values to the vdisk NSDs from the specified recovery group node classes. A failure group FG value must be a whole number from 1 to 65535.

--trim {auto | no}
With the **mmvdisk filesystem create**, **mmvdisk filesystem add**, and **mmvdisk filesystem change** commands, specify if the vdisk NSD trim function is enabled (`auto`) or disabled (no) for capable file system vdisk set member NSDs. The `--trim auto` option is ignored for any targeted vdisk set NSDs that are not capable of vdisk NSD trim. See *IBM Storage Scale RAID: Administration* for a description of the vdisk NSD trim function.

--register
With the **mmvdisk filesystem change** command, register any missing associations between a file system and its vdisk sets. If no change is required, the `--register` option will simply report that the associations between a file system and its vdisk sets are properly registered.

--confirm
Confirms the deletion of an entire or vdiskset of mmvdisk file system. This flag bypasses the interactive prompt that would otherwise be printed to confirm file system deletion.

-Y
Specifies that the **mmvdisk filesystem list** command produce colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk filesystem** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

- To create file system udata by using vdisk set VS1, and to pass the `-Q yes` and `-T /udata` parameters to **mmcrfs**:

```
mmvdisk filesystem create --file-system udata --vdisk-set VS1 --mmcrfs -Q yes -T /udata
```

The system displays output similar to the following:

```
mmvdisk: Creating file system 'udata'.
mmvdisk: The following disks of udata will be formatted on node ess01io1:
mmvdisk:     RG001VS001: size 346212352 MB
mmvdisk:     RG002VS001: size 346212352 MB
mmvdisk: Formatting file system ...
mmvdisk: Disks up to size 2.59 PB can be added to storage pool system.
mmvdisk: Creating Inode File
mmvdisk:    0 % complete on Tue Jun  5 11:36:34 2018
mmvdisk:   11 % complete on Tue Jun  5 11:36:39 2018
mmvdisk:   56 % complete on Tue Jun  5 11:37:05 2018
mmvdisk:   88 % complete on Tue Jun  5 11:37:15 2018
mmvdisk:  100 % complete on Tue Jun  5 11:37:17 2018
mmvdisk: Creating Allocation Maps
mmvdisk: Creating Log Files
mmvdisk:    6 % complete on Tue Jun  5 11:37:23 2018
mmvdisk:  100 % complete on Tue Jun  5 11:37:24 2018
mmvdisk: Clearing Inode Allocation Map
mmvdisk: Clearing Block Allocation Map
mmvdisk: Formatting Allocation Map for storage pool system
mmvdisk:   57 % complete on Tue Jun  5 11:37:36 2018
mmvdisk:  100 % complete on Tue Jun  5 11:37:39 2018
mmvdisk: Completed creation of file system /dev/udata.
```

- To add vdisk set VS2 to file system udata:

```
mmvdisk filesystem add --file-system udata --vdisk-set VS2
```

The system displays output similar to the following:

```
mmvdisk: The following disks of udata will be formatted on node ess01io1:
mmvdisk:     RG001VS002: size 346212352 MB
mmvdisk:     RG002VS002: size 346212352 MB
mmvdisk: Extending Allocation Map
mmvdisk: Checking Allocation Map for storage pool system
mmvdisk:   37 % complete on Tue Jun  5 11:43:06 2018
mmvdisk:   76 % complete on Tue Jun  5 11:43:16 2018
mmvdisk:  100 % complete on Tue Jun  5 11:43:22 2018
mmvdisk: Completed adding disks to file system udata.
```

- To list file system udata and show its vdisk sets and file system NSD information:

```
mmvdisk filesystem list --file-system udata
```

The system displays output similar to the following:

```
                                vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS1             ESS01L              1               1  yes       yes    system
VS1             ESS01R              1               2  yes       yes    system
```

```
VS2            ESS01L             1              1  yes       yes    system
VS2            ESS01R             1              2  yes       yes    system
```

- To delete vdisk set VS2 from file system udata:

```
mmvdisk filesystem delete --file-system udata --vdisk-set VS2
```

The system displays output similar to the following:

```
mmvdisk: This will run the GPFS mmdeldisk command on file system 'udata'
mmvdisk: which may take hours to complete and should not be interrupted.

mmvdisk: Do you wish to continue (yes or no)? yes

mmvdisk: Removing vdisk set 'VS2' from file system 'udata'.
mmvdisk: Deleting disks ...
mmvdisk: Scanning file system metadata, phase 1 ...
mmvdisk: Scan completed successfully.
mmvdisk: Scanning file system metadata, phase 2 ...
mmvdisk: Scan completed successfully.
mmvdisk: Scanning file system metadata, phase 3 ...
mmvdisk: Scan completed successfully.
mmvdisk: Scanning file system metadata, phase 4 ...
mmvdisk: Scan completed successfully.
mmvdisk: Scanning user file metadata ...
mmvdisk:  100.00 % complete on Tue Jun  5 12:30:54 2018 (503808 inodes with total 13872 MB data
processed)
mmvdisk: Scan completed successfully.
mmvdisk: Checking Allocation Map for storage pool system
mmvdisk:   11 % complete on Tue Jun  5 12:30:59 2018
mmvdisk:   41 % complete on Tue Jun  5 12:31:09 2018
mmvdisk:   77 % complete on Tue Jun  5 12:31:19 2018
mmvdisk:  100 % complete on Tue Jun  5 12:31:24 2018
mmvdisk: tsdeldisk completed.
```

- To delete the entire file system udata and bypass the interactive confirmation prompt:

```
mmvdisk filesystem delete --file-system udata --confirm
```

The system displays output similar to the following:

```
mmvdisk: All data on the following disks of udata will be destroyed:
mmvdisk:     RG001VS001
mmvdisk:     RG002VS001
mmvdisk: Completed deletion of file system /dev/udata.
```

- Suppose vdisk set VS01 is defined in two scale-out recovery groups, RG01 and RG02. The node class for RG01 is NC01 and the node class for RG02 is NC02. To create file system FS01 using vdisk set VS01, and to assign failure group 100 to the vdisk NSDs in RG01 (node class NC01) and failure group 200 to the vdisk NSDs in RG02 (node class NC02):

```
mmvdisk filesystem create --file-system FS01 --vdisk-set VS01 --failure-groups
NC01=100,NC02=200
```

The system displays output similar to the following:

```
mmvdisk: Creating file system 'FS01'.
mmvdisk: The following disks of FS01 will be formatted on node server07:
mmvdisk:     RG001LG001VS001: size 8064 MB
mmvdisk:     RG001LG002VS001: size 8064 MB
mmvdisk:     RG001LG003VS001: size 8064 MB
mmvdisk:     RG001LG004VS001: size 8064 MB
mmvdisk:     RG001LG005VS001: size 8064 MB
mmvdisk:     RG001LG006VS001: size 8064 MB
mmvdisk:     RG001LG007VS001: size 8064 MB
mmvdisk:     RG001LG008VS001: size 8064 MB
mmvdisk:     RG001LG009VS001: size 8064 MB
mmvdisk:     RG001LG010VS001: size 8064 MB
mmvdisk:     RG001LG011VS001: size 8064 MB
mmvdisk:     RG001LG012VS001: size 8064 MB
mmvdisk:     RG002LG001VS001: size 8064 MB
mmvdisk:     RG002LG002VS001: size 8064 MB
mmvdisk:     RG002LG003VS001: size 8064 MB
```

```
mmvdisk:     RG002LG004VS001: size 8064 MB
mmvdisk:     RG002LG005VS001: size 8064 MB
mmvdisk:     RG002LG006VS001: size 8064 MB
mmvdisk:     RG002LG007VS001: size 8064 MB
mmvdisk:     RG002LG008VS001: size 8064 MB
mmvdisk:     RG002LG009VS001: size 8064 MB
mmvdisk:     RG002LG010VS001: size 8064 MB
mmvdisk:     RG002LG011VS001: size 8064 MB
mmvdisk:     RG002LG012VS001: size 8064 MB
mmvdisk: Formatting file system ...
mmvdisk: Disks up to size 269.62 GB can be added to storage pool system.
mmvdisk: Creating Inode File
mmvdisk:   57 % complete on Wed May  8 07:18:54 2019
mmvdisk:   85 % complete on Wed May  8 07:18:59 2019
mmvdisk:  100 % complete on Wed May  8 07:18:59 2019
mmvdisk: Creating Allocation Maps
mmvdisk: Creating Log Files
mmvdisk:    3 % complete on Wed May  8 07:19:06 2019
mmvdisk:  100 % complete on Wed May  8 07:19:08 2019
mmvdisk: Clearing Inode Allocation Map
mmvdisk: Clearing Block Allocation Map
mmvdisk: Formatting Allocation Map for storage pool system
mmvdisk: Completed creation of file system /dev/FS01.
```

- To verify that failure group 100 is used in RG01 and that failure group 200 is used in RG02:

```
mmvdisk filesystem list --file-system FS01
```

The system displays output similar to the following:

```
                                vdisk     list of       holds    holds  storage
vdisk set       recovery group  count  failure groups  metadata  data    pool
--------------  --------------  -----  --------------  --------  -----  -------
VS01            RG01               12             100  yes       yes    system
VS01            RG02               12             200  yes       yes    system
```

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk pdisk command” on page 388
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk nodeclass command” on page 346

Also, see the following commands in the *IBM Storage Scale: Command and Programming Reference* guide:

- *mmadddisk command*
- *mmdeldisk command*
- *mmlsdisk command*
- *mmchfs command*
- *mmcrfs command*
- *mmdelfs command*
- *mmlsfs command*

Location

`/usr/lpp/mmfs/bin`

mmvdisk vdiskset command

Manages mmvdisk vdisk sets for IBM Storage Scale RAID.

Synopsis

```
mmvdisk vdiskset define  --vdisk-set VdiskSet
                         --recovery-group {all | RgName[,RgName...]}
                         --code RaidCode  --block-size BlockSize
                         --set-size {n% | n | nK | nM | nG | nT}
                         [--declustered-array DaName]
                         [--nsd-usage NsdUsage [--storage-pool StoragePool]]
```

or

```
mmvdisk vdiskset define  --vdisk-set VdiskSet
                         --recovery-group  RgName[,RgName...]
                         [--force-incompatible]
```

or

```
mmvdisk vdiskset define  --vdisk-set NewVdiskSet --copy ExistingVdiskSet
                         [--recovery-group  RgName[,RgName...]]
                         [--declustered-array DaName]
                         [--force-incompatible]
```

or

```
mmvdisk vdiskset undefine  --vdisk-set VdiskSet
                         --recovery-group  RgName[,RgName...]
```

or

```
mmvdisk vdiskset undefine  --vdisk-set VdiskSet [--confirm]
```

or

```
mmvdisk vdiskset create  --vdisk-set {all | VdiskSet[,VdiskSet...]}
```

or

```
mmvdisk vdiskset delete --vdisk-set {all | VdiskSet[,VdiskSet...]} |
                       [--recovery-group  RgName[,RgName...]] [-p]
```

or

```
mmvdisk vdiskset rename --vdisk-set VdiskSet --new-name NewName
```

or

```
mmvdisk vdiskset list [--vdisk-set {all | VdiskSet[,VdiskSet...]}] [-Y]
```

or

```
mmvdisk vdiskset list --recovery-group {all | RgName[,RgName...]} [-Y]
                     [--declustered-array DaName[,DaName...]]
```

or

```
mmvdisk vdiskset list --file-system {all | FsName[,FsName...]} [-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk vdiskset** command to define, size, and create uniform vdisk NSDs across IBM Storage Scale RAID recovery groups. The vdisk sets that result can then be used as units in constructing IBM Storage Scale file systems.

With **mmvdisk**, user vdisks are managed collectively as vdisk sets: vdisk NSDs of identical attributes from one or more recovery groups, with each log group of each recovery group contributing one member vdisk NSD to the vdisk set. To create a filesystem balanced across several recovery groups, a vdisk set is defined using those recovery groups. The vdisk set definition provides a preview of the sizing impact of the vdisk set on the recovery groups and the servers. If the sizing is considered acceptable, the vdisk set can be created, meaning that the individual member vdisks and NSDs in each recovery group are all created. The created vdisk set can be used to create an IBM Storage Scale RAID file system (or be added to an existing IBM Storage Scale RAID file system). All the member vdisk NSDs of a vdisk set must belong to the same file system. A file system can contain one or more vdisk sets.

The **mmvdisk vdiskset define** command is used to define vdisk sets, to add a recovery group to an existing vdisk set definition, or to define a new vdisk set using a copy of an existing vdisk set definition.

The definition of a vdisk set must include:

- A vdisk set name.
- One or more recovery groups.
- The name of a single declustered array, which must be present in each of the recovery groups. The member vdisk NSDs will be sized against and created in this declustered array. If each recovery group has only the single user declustered array, "DA1," the name need not be specified.
- The RAID code used by the member vdisk NSDs. Accepted RAID codes are 3WayReplication, 4WayReplication, 4+2p, 4+3p, 8+2p, and 8+3p.
- The block size used by the member vdisk NSDs. The block size is constrained by the RAID code. Valid values for 3WayReplication and 4WayReplication are 256k, 512k, 1m, and 2m. Valid values for 4+2P and 4+3P are 512k, 1m, 2m, 4m, 8m. Valid values for 8+2P and 8+3P are 512k, 1m, 2m, 4m, 8m, 16m.
- The desired aggregate size of the vdisk set members in one recovery group. This set size can be specified as a percentage (whole numbers from 1% to 100% using the % suffix) or as a number of bytes (a number, optionally followed by one of the base 2 suffixes K, M, G, or T). If the vdisk set size is given as a percentage, it specifies the raw size to use from the declustered array, including RAID code redundancy. If the vdisk set size is given as a number of bytes, it specifies the desired usable size of the vdisk set, excluding RAID code redundancy. The vdisk set size is used to calculate the usable size of a single vdisk NSD member of the vdisk set in one recovery group. It is this calculated usable size that becomes part of the vdisk set definition, so if the size of a declustered array should ever change, the size of the individual member vdisk NSDs remains constant. Note that the resulting actual set size will almost certainly differ from the requested set size, especially if a number of usable bytes is specified, since the IBM Storage Scale RAID internal data structures to accommodate the requested size will depend on the number and size of the pdisks and on the RAID code and block size of the member vdisks.
- The NSD usage of the vdisk set. This is the IBM Storage Scale file system data usage for the vdisk NSDs. Valid values are dataAndMetadata, metadataOnly, and dataOnly. The default is dataAndMetadata.
- The file system storage pool for the vdisk set. This is the IBM Storage Scale file system storage pool for the vdisk NSDs. If the NSD usage is dataAndMetadata or metadataOnly, the storage pool must be "system" and need not be specified. If the NSD usage is dataOnly, the storage pool must be specified and cannot be "system."

When multiple recovery groups are used in an initial vdisk set definition, the declustered arrays must have perfectly compatible characteristics (same number and size of pdisks, number of spare pdisks, or pdisk hardware type, etc.). This ensures that the member vdisk NSDs in the vdisk set will have the same

characteristics, especially that the individual and aggregate size of the member vdisk NSDs will be the same in each recovery group.

There is a special vdisk set block size consideration for declustered arrays built from non-rotational SSD or NVMe pdisks. Non-rotational pdisk devices perform more slowly with the largest block sizes. For the 3WayReplication and 4WayReplication RAID codes with non-rotational devices, the block size should be 1 MiB or less. For the 4+2P and 4+3P RAID codes with non-rotational devices, the block size should be 2 MiB or less. For the 8+2P and 8+3P RAID codes with non-rotational devices, the block size should be 4 MiB or less. The choice of block size for non-rotational devices can otherwise be made independently of this special consideration. The **mmvdisk vdiskset define** command will allow the larger block sizes for non-rotational declustered arrays, but will print a notification that performance will be affected.

To add recovery groups to an existing vdisk set definition, use **mmvdisk vdiskset define** with only the `--vdisk-set` and `--recovery-group` parameters. The declustered array in the added recovery groups must be compatible with the declustered array that was used in the initial vdisk set definition. In carefully considered circumstances, it is possible to force a vdisk set definition into an incompatible declustered array using the `--force-incompatible` flag.

A vdisk set can be defined by copying the definition of an existing vdisk set using the `--copy` flag. This can be useful when it is desired to duplicate the exact size of existing file system vdisk NSDs in recovery groups that have been resized with the `mmvdisk recoverygroup resize` command. The exact member size and all attributes of the copied vdisk set definition will be unchanged from the original vdisk set definition, except if the `--declustered-array` or `--recovery-group` flags are given to specify a different declustered array or different recovery groups in the copy. In carefully considered circumstances, it is possible to copy a vdisk set definition into an incompatible declustered array using the `--force-incompatible` flag.

When a vdisk set is defined, **mmvdisk** will show the space used by the definition in both the declustered array and in the recovery group server memory. This permits the administrator to evaluate the definition in context before committing to its creation, or to decide to undefine it and try a different definition.

A vdisk set can be undefined using the **mmvdisk vdiskset undefine** command. This can be used to remove individual recovery groups from a vdisk set, or to remove the entire vdisk set definition. Any member vdisk NSDs in the affected recovery groups must first have been deleted using the **mmvdisk vdiskset delete** command.

The member vdisk NSDs of a vdisk set definition are created using the **mmvdisk vdiskset create** command. A vdisk set is "created" when all the actual member vdisk NSDs exist in each recovery group in the vdisk set definition. The **mmvdisk vdiskset create** command only creates those vdisk NSDs that do not already exist; this permits a vdisk set to be defined and created, then to have another recovery group added and then to create only the vdisk NSDs for the added recovery group. Only created vdisk sets can be used with the **mmvdisk filesystem** command to add to or create IBM Storage Scale RAID file systems.

Member vdisk NSDs can be deleted from a vdisk set using the **mmvdisk vdiskset delete** command. The member vdisk NSDs to be deleted can be limited to a recovery group, or can include the entire vdisk set. In either case, it is not permitted to delete member vdisk NSDs that are in a file system.

Use the **mmvdisk vdiskset rename** command to change the name of a vdisk set. This is useful for giving meaningful names to the vdisk sets defined by the **mmvdisk recoverygroup convert** command.

Use the **mmvdisk vdiskset list** command to list vdisk sets and to show their attributes and sizing.

Parameters

mmvdisk vdiskset define
: Defines vdisk sets across IBM Storage Scale RAID recovery groups.

mmvdisk vdiskset undefine
: Remove vdisk set definitions from IBM Storage Scale RAID recovery groups.

mmvdisk vdiskset create
Create the member vdisk NSDs of a vdisk set.

mmvdisk vdiskset delete
Delete member vdisk NSDs from a vdisk set. The member vdisk NSDs must not belong to a file system.

mmvdisk vdiskset rename
Rename a vdisk set.

mmvdisk vdiskset list
List vdisk sets and show their attributes and sizing.

--vdisk-set ***VdiskSet***
Specifies a single vdisk set to define, undefine, or rename.

--vdisk-set {all | VdiskSet[,VdiskSet...]}
Specifies vdisk sets to create, delete, or list.

--recovery-group {all | ***RgName*****[,*****RgName*****...]}**
Specifies recovery groups in which vdisk sets are to be defined or listed.

--recovery-group ***RgName*****[,*****RgName*****...]**
Specifies recovery groups from which vdisk NSDs are to be deleted, or in which a vdisk set is to be defined or undefined.

--code ***RaidCode***
Specifies the RAID code of a vdisk set definition.

--block-size ***BlockSize***
Specifies the block size of a vdisk set definition.

--set-size {*n***% |** ***n*** **|** ***n*****K |** ***n*****M |** ***n*****G |** ***n*****T}**
Specifies the desired set size, within a single recovery group, of a vdisk set definition. Note that the resulting actual set size will almost certainly differ from the requested set size, especially if a number of usable bytes is specified, since the IBM Storage Scale RAID internal data structures to accommodate the requested size will depend on the number and size of the pdisks and on the RAID code and block size of the member vdisks.

--declustered-array ***DaName***
Specifies the declustered array of a vdisk set definition.

--nsd-usage ***NsdUsage***
Specifies the file system NSD usage of a vdisk set definition.

--storage-pool ***StoragePool***
Specifies the file system storage pool of a vdisk set definition.

--force-incompatible
When extending or copying a vdisk set definition to additional recovery groups, attempts to ignore incompatible declustered array characteristics. This option should be used only under careful consideration, as vdisk NSDs from incompatible declustered arrays can lead to unbalanced file system performance.

--copy ***ExistingVdiskSet***
When defining a new vdisk set as a copy of an existing vdisk, specifies the existing vdisk set from which the attributes are copied.

--new-name ***NewName***
With the `mmvdisk vdiskset rename` command, specifies the new name for a vdisk set.

--declustered-array ***DaName*****[,*****DaName*****...]**
With the **mmvdisk vdiskset list** command, choose the declustered arrays in which vdisk sets are to be listed.

--file-system {all | ***FsName*****[,*****FsName*****...]}**
With the **mmvdisk vdiskset list** command, choose file systems from which vdisk sets are to be listed.

--confirm
Confirms the removal of a vdisk set definition. This flag bypasses the interactive prompt that would otherwise be printed to confirm undefining a vdisk set.

-p
With the **mmvdisk vdiskset delete** command, the -p flag indicates that the member vdisk NSDs are in a recovery group that is "permanently damaged" and must be deleted without access to the recovery group. This option is not permitted if the recovery group is active. This is an unusual maintenance case that should only be performed under the direction of the IBM Service. Before a vdisk set can be deleted using -p, the vdisk set must first be deleted from any file system.

-Y
Specifies that the **mmvdisk vdiskset list** command produce colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmvdisk vdiskset** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Example

To see the available space for defining vdisk sets in a newly created recovery group pair:

```
mmvdisk vdiskset list --recovery-group ESS01L,ESS01R
```

The system displays output similar to the following:

```
                declustered                 capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
ESS01L          DA1          HDD     911 TiB   911 TiB   100%  -
ESS01R          DA1          HDD     911 TiB   911 TiB   100%  -

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB   387 MiB  -
```

This shows that no vdisk sets are yet defined in either recovery group; the total free raw disk capacity in each declustered array; and the available memory for vdisk set maps on the servers in the recovery group pair node class.

Example

To define vdisk set VS1 in recovery groups ESS01L and ESS01R using RAID code 8+3p, an 8 MiB block size, and 50% of the available raw space:

```
mmvdisk vdiskset define --vdisk-set VS1 --recovery-group ESS01L,ESS01R --code 8+3p --block-size 8m --set-size
50%
```

The system displays output similar to the following:

```
mmvdisk: Vdisk set 'VS1' has been defined.
mmvdisk: Recovery group 'ESS01L' has been defined in vdisk set 'VS1'.
mmvdisk: Recovery group 'ESS01R' has been defined in vdisk set 'VS1'.
```

```
                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  -----  -------- --------  -------  -------------------------
VS1                 2  330 TiB  455 TiB  no       -, DA1, 8+3p, 8 MiB, dataAndMetadata, system

                declustered                capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
ESS01L          DA1          HDD     911 TiB   455 TiB    50%  VS1
ESS01R          DA1          HDD     911 TiB   455 TiB    50%  VS1

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
ESS01          29 GiB  9227 MiB  VS1 (8839 MiB)
```

This output shows the attributes of the newly defined vdisk set; that it is not yet created; and the space and memory sizing demands it places on the declustered arrays and the servers in the recovery group node class. Note that, because ESS01L and ESS01R are paired recovery groups with just one log group each, the count of the member vdisks in vdisk set VS1 is 2, one per log group (equivalently, one per paired recovery group).

Example

To create the member vdisk NSDs in vdisk set VS1:

```
mmvdisk vdiskset create --vdisk-set VS1
```

The system displays output similar to the following:

```
mmvdisk: 2 vdisks and 2 NSDs will be created in vdisk set 'VS1'.
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001VS001
mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002VS001
mmvdisk: Created all vdisks in vdisk set 'VS1'.
mmvdisk: (mmcrnsd) Processing disk RG001VS001
mmvdisk: (mmcrnsd) Processing disk RG002VS001
mmvdisk: Created all NSDs in vdisk set 'VS1'.
```

Example

To see the available space for defining vdisk sets in a newly created scale-out recovery group that has 6 servers:

```
mmvdisk vdiskset list --recovery-group RG01
```

The system displays output similar to the following:

```
                declustered                capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
RG01            DA1          NVMe   3827 GiB  3827 GiB   100%  -
RG01            DA2          HDD    5289 GiB  5289 GiB   100%  -

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
NC01         3276 MiB   384 MiB  -
```

Example

To define vdisk set VS1DA2 in declustered array DA2 of recovery group RG01 using RAID code 4+2p, a 2 MiB block size, and 50% of the available raw space:

```
mmvdisk vdiskset define --vdisk-set VS1DA2 --recovery-group RG01 --declustered-array DA2 --code
```

```
4+2p
--block-size 2m --set-size 50%
```

The system displays output similar to the following:

```
mmvdisk: Vdisk set 'VS1DA2' has been defined.
mmvdisk: Recovery group 'RG01' has been defined in vdisk set 'VS1DA2'.

                     member vdisks
vdisk set       count   size   raw size  created  file system and attributes
--------------  ----- -------- --------  -------  -------------------------
VS1DA2             12  145 GiB  219 GiB  no       -, DA2, 4+2p, 2 MiB, dataAndMetadata, system

                declustered                capacity           all vdisk sets defined
recovery group     array     type  total raw  free raw  free%  in the declustered array
--------------  -----------  ----  ---------  --------  -----  ------------------------
RG01            DA2          HDD    5289 GiB  2652 GiB    50%  VS1DA2

                  vdisk set map memory per server
node class  available  required  required per vdisk set
----------  ---------  --------  ----------------------
NC01         3276 MiB   393 MiB  VS1DA2 (9216 KiB)
```

This output shows the attributes of the newly defined vdisk set; that it is not yet created; and the space and memory sizing demands it places on the declustered array and the servers in the recovery group node class. Note that, because RG01 is a scale-out recovery group with 6 servers and therefore 12 log groups, the count of the member vdisks in vdisk set VS1DA2 is 12, one per log group (equivalently, two per scale-out recovery group server).

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk pdisk command” on page 388
- “mmvdisk vdisk command” on page 386
- “mmvdisk nodeclass command” on page 346

Location

`/usr/lpp/mmfs/bin`

mmvdisk vdisk command

Lists individual IBM Storage Scale RAID vdisks.

Synopsis

```
mmvdisk vdisk list  [--vdisk-set {all | VsName[,VsName...]} [-Y]
```

or

```
mmvdisk vdisk list  --file-system FsName [-Y]
```

or

```
mmvdisk vdisk list  --recovery-group {all | RgName[,RgName...]}
                    [--vdisk-set {all | VsName[,VsName...]}]
                    [--declustered-array DaName[,DaName...]]
```

```
                     [--log] [-Y]
```

or

```
mmvdisk vdisk list  --vdisk VdiskName[,VdiskName...] [-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the `mmvdisk vdisk` command to list individual IBM Storage Scale RAID vdisks.

Because **mmvdisk** administration of IBM Storage Scale RAID creates and deletes individual vdisks either as members of vdisk sets or as recovery group log vdisks, the `mmvdisk vdisk` command only supports the listing of individual vdisks.

Vdisks can be selected based on membership by vdisk set, by file system, or by recovery group and declustered array. Individual vdisks can also be listed by name.

By default, only user vdisks are listed - those vdisks that are in file systems or are eligible to be in file systems.

Log vdisks can be listed using the `--log` flag when a recovery group is specified.

Parameters

mmvdisk vdisk list
: List selected individual vdisks.

--vdisk-set ***all | VsName[,VsName...]***
: Specifies the vdisk sets from which member vdisks are to be listed.

--file-system ***FsName***
: Specifies the file system from which vdisks are to be listed.

--recovery-group ***all | RgName[,RgName...]***
: Specifies the recovery groups from which vdisks are to be listed.

--declustered-array ***DaName[,DaName...]***
: Restricts the listing of recovery group vdisks to the specified declustered arrays. If omitted, vdisks are listed from all declustered arrays in a recovery group.

--log
: Restricts the listing of recovery group vdisks to log vdisks. If omitted, only user vdisks are listed.

-Y
: Specifies that the `mmvdisk vdisk list` command produce colon-delimited raw output.

Exit status

0
: Successful completion.

nonzero
: A failure occurred.

Security

You must have root authority to run the **mmvdisk vdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Example

To list the user vdisks that are in file system FS3:

```
mmvdisk vdisk list --file-system FS3
```

The system displays output similar to the following:

```
                                                                declustered array, RAID code,
vdisk               vdisk set       file system  recovery group            block size
remarks
------------------  --------------  -----------  --------------  ------------------------------
-------
RG001VS001          VS3             FS3          ESS01L          DA1, 8+3p, 8 MiB
RG002VS001          VS3             FS3          ESS01R          DA1, 8+3p, 8 MiB
RG003VS001          VS3             FS3          ESS02L          DA1, 8+3p, 8 MiB
RG004VS001          VS3             FS3          ESS02R          DA1, 8+3p, 8 MiB
```

Example

To list the log vdisks in recovery group ESS01L:

```
mmvdisk vdisk list --log --recovery-group ESS01L
```

The system displays output similar to the following:

```
                                                    declustered array, RAID code,
vdisk               vdisk set       file system  recovery group           block size           remarks
------------------  --------------  -----------  --------------  ---------------------------  -------
RG001LOGHOME        -               -            ESS01L          DA1, 4WayReplication, 2 MiB  log home
RG001LLOGTIP        -               -            ESS01L          NVR, 2WayReplication, 2 MiB  log tip
RG001LLOGTIPBACKUP  -               -            ESS01L          SSD, Unreplicated, 2 MiB     log tip backup
```

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk nodeclass command” on page 346
- “mmvdisk pdisk command” on page 388

Location

/usr/lpp/mmfs/bin

mmvdisk pdisk command

Manages **mmvdisk** recovery group pdisks for IBM Storage Scale RAID.

Synopsis

```
mmvdisk pdisk list --recovery-group {all | RgName[,RgName]}
                   [--declustered-array DaName | --pdisk PdiskName]
                   [--replace | --not-ok | --ssd-endurance-percentage N]  [-Y]
```

or

```
mmvdisk pdisk list  -L
                   --recovery-group {{all | RgNaame}  [--declusterebd-array DaName |  --pdisk
PdiskName]
```

```
                    [--replace | --not-ok | --ssd-endurance-percentage N][-Y]
```

or

```
mmvdisk pdisk replace  --prepare
                       --recovery-group RgName
                       {--pdisk PdiskName | --location LocationCode}
                       [--force-rg] [--force]
```

or

```
mmvdisk pdisk replace --recovery-group RgName
                       {--pdisk PdiskName | --location LocationCode}
                       [--force-rg] [--force-fru] [-v {yes | no}]
```

or

```
mmvdisk pdisk replace --cancel --recovery-group RgName
                      {--pdisk PdiskName | --location LocationCode}
                      [--force-rg]
```

or

```
mmvdisk pdisk change --recovery-group RgName
                     --pdisk PdiskName
                     [--identify {on | off}]
                     [--clear-error-counters]
                     [--diagnose |
                     --suspend |
                     --resume |
                     --revive |
                     --revive-failing |
                     --revive-slow |
                     --begin-service-drain |
                     --end-service-drain |
                     --simulate-dead |
                     --simulate-failing]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk pdisk** command to manage the pdisks in IBM Storage Scale RAID recovery groups.

Attention: The `mmvdisk pdisk change` command is a low-level command that should only be used in extreme situations under the guidance of IBM Service.

Use the `mmvdisk pdisk list` command to show the pdisk information for selected pdisks. Pdisks can be selected for listing based on recovery group, declustered array, or pdisk maintenance state. The `--not-ok` flag will list pdisks that are not in service. The `--replace` flag will list pdisks that are marked for replacement. The `--ssd-endurance-percentage` *N* flag will list SSD pdisks with write endurance percentage greater than or equal to *N*.

By default, the `mmvdisk pdisk list` command prints a one-line summary for each listed pdisk. If the `-L` flag is used, pdisks are listed in stanza format, which shows detailed pdisk attributes and is quite voluminous.

Use the `mmvdisk pdisk replace` command to perform pdisk replacement when an IBM Storage Scale RAID recovery group is reported to need service.

Replacing a pdisk requires the following three steps:

1. Run the `mmvdisk pdisk replace --prepare` command to prepare the pdisk for physical removal.
2. Physically remove the disk and replace it with a new disk of the same type.

3. Run the `mmvdisk pdisk replace` command to complete the replacement.

Note: New disks take the name of the replaced pdisks. In the event that replaced pdisks have not completely drained, they will be given a temporary name consisting of the old pdisk name with a suffix of the form #nnnn. The temporary pdisk will have the `adminDrain` pdisk state flag set and will be deleted once drained. For example, a pdisk named `e1s05` will receive a temporary name similar to `e1s05#0010` when the `adminDrain` state flag is set. This allows the new disk that is replacing it to be named `e1s05` immediately rather than waiting for the old disk to be completely drained and deleted. Until the draining and deleting process completes, both the new pdisk `e1s05` and the old pdisk `e1s05#0010` will appear in the listing of the `mmvdisk pdisk list` and `mmvdisk recoveryroup list --pdisk` commands.

The `mmvdisk pdisk replace --cancel` command can be used to cancel the replacement of a pdisk that has been prepared for replacement. Canceling a prepared replacement does not render a required replacement unnecessary.

The `mmvdisk pdisk change` command changes the runtime state of a pdisk. Incorrect use of this command can cause file system or recovery group data to become unavailable or even unrecoverable. It should only be used in extreme situations under the guidance of IBM Service.

Parameters

`mmvdisk pdisk list`
List pdisks.

`mmvdisk pdisk replace`
Replace failed pdisks.

`mmvdisk pdisk change`
Change pdisk states. This is a low-level command that should only be used in extreme situations under the guidance of IBM service.

`--recovery-group` ***all | RgName[,RgName...]***
With the `mmvdisk pdisk list` command, specifies the recovery groups from which to list pdisks.

`--declustered-array` ***DaName***
Restricts pdisk listing to the specified declustered array.

`--pdisk` ***PdiskName***
Specifies a pdisk by name.

`--replace`
Specifies that only pdisks that are marked for replacement are to be listed.

`--not-ok`
Specifies that only pdisks that are not functioning correctly are to be listed.

`--ssd-endurance-percentage` ***N***
Specifies that only SSD pdisks with write endurance percentage greater than or equal to *N* are to be listed. Valid values for *N* are whole numbers from 0 to 100.

`--recovery-group` ***RgName***
With the `mmvdisk pdisk replace` and `mmvdisk pdisk change` commands, specifies the recovery group for the target pdisk.

`--prepare`
With the `mmvdisk pdisk replace` command, prepares a pdisk for removal and replacement.

`--cancel`
With the `mmvdisk pdisk replace` command, cancels the preparation of a pdisk for removal and replacement.

`--location` ***LocationCode***
With the `mmvdisk pdisk replace` command, specifies a pdisk by location code. Pdisk location codes can be found in the long form stanza output of the `mmvdisk pdisk list` command.

`--force`
With the `mmvdisk pdisk replace --prepare` command, prepares a working pdisk that is not marked for replacement. The `--force` flag does not permit the replacement of an otherwise good

pdisk. It merely suspends the pdisk from use for temporary removal, inspection, and reseating of the same exact physical disk; a different disk must not be inserted. To resume operation of a good pdisk that was suspended with `--force`, only the `mmvdisk pdisk replace --cancel` command is permitted.

`--force-rg`
With the `mmvdisk pdisk replace` command, indicates that the pdisk being replaced is in a multi-disk carrier that also contain pdisks that are not in a recovery group, or in a different recovery group.

`--force-fru`
With the `mmvdisk pdisk replace` command, permits the replacement disk to have a different FRU (or type) than the removed pdisk.

`-v {`*yes* **| *no*`}`**
With the `mmvdisk pdisk replace` command, specifies whether the replacement disk should be verified to be empty of IBM Storage Scale RAID data before it is incorporated into the recovery group. The default is *yes*, verify that the pdisk is empty of data.

`--identify {`*on* **| *off*`}`**
Specifies that the identify light for the pdisk should be turned on or `off`.

`--clear-error-counters`
Specifies that the pdisk error counters for the pdisk should be cleared.

`--diagnose`
Runs basic tests on the pdisk. If no problems are found, the pdisk state automatically returns to ok.

`--suspend`
Suspends I/O to the pdisk until a subsequent `--resume` command is given. If a pdisk remains in the suspended state for longer than a predefined timeout period, IBM Storage Scale RAID begins rebuilding the data from that pdisk into spare space.

Attention: This option is to be used with caution and only when instructed to perform disk maintenance manually, bypassing the automatic system provided by the `mmvdisk pdisk replace` command.

If a pdisk is removed using this option, vdisks that store data on the removed pdisk will become temporarily degraded, requiring data that was stored on the removed pdisk to be rebuilt from redundancy. Also, if you try to remove more pdisks than the redundancy level of the least redundant vdisk in that declustered array, data will become inaccessible. Therefore, when preparing to remove a pdisk, use the `--begin-service-drain` and `--end-service-drain` options instead of this option.

`--resume`
Cancels a previously given `--suspend` command and resumes use of the pdisk.

Use this option only when instructed to perform disk maintenance manually, bypassing the automatic system provided by the `mmvdisk pdisk replace` command.

`--revive`
Attempts to make an out-of-service pdisk usable again by removing dead, failing, and readonly pdisk state flags. Data allocated on the disk that has not been rebuilt onto spare space can become readable again; however, any data that has already been reported as lost cannot be recovered.

`--revive-failing`
Like the `--revive` parameter, except that additional diagnostics particular to the failing pdisk state flag are attempted.

`--revive-slow`
Like the `--revive` parameter, except that additional diagnostics particular to the slow pdisk state flag are attempted.

`--begin-service-drain`
Starts draining the pdisk so that it can be temporarily removed. After issuing the command with this option, wait until the pdisk has been completely drained before removing the pdisk.

Note: This process requires that there be sufficient spare space in the declustered array for the data that is to be drained.

--end-service-drain

Starts returning drained data to a pdisk after it has been brought back online.

--simulate-dead

Forces the failure of a pdisk by setting the `simulatedDead` pdisk state flag.

Attention: This option must be used with caution. If the total number of failures in a declustered array exceeds the fault tolerance of any vdisk in that array, permanent data loss might result.

--simulate-failing

Forces the failure of a pdisk by setting the `simulatedFailing` pdisk state flag.

Attention: This option must be used with caution. If the total number of failures in a declustered array exceeds the fault tolerance of any vdisk in that array, permanent data loss might result.

-L

Specifies that the `mmvdisk pdisk list` command produce long form stanza output.

-Y

Specifies that the `mmvdisk pdisk list` command produce colon-delimited raw output.

Exit status

0

Successful completion.

nonzero

A failure occurred.

Security

You must have root authority to run the **mmvdisk pdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Example

To list the pdisks in recovery group ESS01L:

```
mmvdisk pdisk list --recovery-group ESS01L
```

The system displays output similar to the following:

```
                          declustered
recovery group  pdisk        array     paths  capacity  free space  FRU (type)       state
--------------  ------------ ----------- -----  --------  ----------  ---------------  -----
ESS01L          e1d1s01      DA1             2  5588 GiB     216 GiB  38L6721          ok
ESS01L          e1d1s02      DA1             2  5588 GiB     216 GiB  38L6721          ok
ESS01L          e1d1s04      DA1             2  5588 GiB     216 GiB  38L6721          ok
ESS01L          e1d1s05      DA1             2  5588 GiB     216 GiB  38L6721          ok
...
```

If a drive is locked because of I/O errors and the drive is not unlocked automatically, the drive state is displayed as SEDLocked.

Example

To show all pdisks marked for replacement in all recovery groups:

```
mmvdisk pdisk list --replace --recovery-group all
```

The system displays output similar to the following:

```
recovery group  pdisk         priority  FRU (type)       location
--------------  ------------  --------  ---------------  --------
ESS01L          e3d4s06           5.78  38L6721          Enclosure SV50919775 Drawer 4 Slot 6
ESS01L          e6d1s02           5.78  38L6721          Enclosure SV50918970 Drawer 1 Slot 2

mmvdisk: A lower priority value means a higher need for replacement.
```

Example

To prepare pdisk e6d1s02 in recovery group ESS01L for replacement:

```
mmvdisk pdisk replace --prepare --recovery-group ESS01L --pdisk e6d1s02
```

The system displays output similar to the following:

```
mmvdisk: Suspending pdisk e6d1s02 of RG ESS01L in location SV50918970-1-2.
mmvdisk: Location SV50918970-1-2 is Enclosure SV50918970 Drawer 1 Slot 2.
mmvdisk: Carrier released.
mmvdisk:
mmvdisk:   - Remove carrier.
mmvdisk:   - Replace disk in location SV50918970-1-2 with type '38L6721'.
mmvdisk:   - Reinsert carrier.
mmvdisk:   - Issue the following command:
mmvdisk:
mmvdisk:   mmvdisk pdisk replace --recovery-group ESS01L --pdisk 'e6d1s02'
```

Example

To replace pdisk e6d1s02 of recovery group ESS01L after a new disk has been inserted:

```
mmvdisk pdisk replace --recovery-group ESS01L --pdisk e6d1s02
```

The system displays output similar to the following:

```
mmvdisk:
mmvdisk: Preparing a new pdisk for use may take many minutes.
mmvdisk:
mmvdisk: The following pdisks will be formatted on node ess01io1:
mmvdisk:       /dev/sdrk
mmvdisk:
mmvdisk: Location SV50918970-1-2 is Enclosure SV50918970 Drawer 1 Slot 2.
mmvdisk: Pdisk e6d1s02 of RG ESS01L successfully replaced.
mmvdisk: Resuming pdisk e6d1s02#047 of RG ESS01L.
mmvdisk: Carrier resumed.
```

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk nodeclass command” on page 346

Location
/usr/lpp/mmfs/bin

mmvdisk sed command

Manages IBM Storage Scale RAID self-encrypting drives (SED).

Synopsis

```
mmvdisk sed enroll --recovery-group RgName --rkmid RKMid --key-uuid KeyId [--confirm]
```

or

```
mmvdisk sed rekey  --recovery-group RgName --rkmid RKMid --key-uuid KeyId [--confirm]
```

or

```
mmvdisk sed list  {--all | --recovery-group RgName[,RgName...] |
                     --recovery-group RgName [--pdisk pdiskname] |
                     --pdisk-path pdisk-path} [-Y]
```

or

```
mmvdisk sed verify  {--all | --recovery-group RgName[,RgName...] |
                       --recovery-group RgName [--pdisk pdiskname] |
                       --pdisk-path pdisk-path} [-Y]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk sed** command to manage SEDs. The command sets a new authentication key (master encryption key), changes the authentication key (MEK) to a new key, manages the encryption of the data on SEDs and locks the drives automatically after a power recycle.

The **mmvdisk sed** command can be run from any ESS I/O node in an IBM Storage Scale cluster. The RKM server, such as GKLM server, must be set up before issuing these commands. The GNR I/O nodes must have direct network access to the RKM server.

Note: The MEK is important and critical information. If it is lost, the access to the encrypted data is also lost permanently. It is recommended to back up the keys by following the remote key manager (RKM) backup procedures.

Parameters

enroll
: Uses a master encryption key (MEK) from the RKM server and configures all the SEDs to enable encryption.

  --rkmid ***RKMiD***
  : Specifies a new RKM ID.

  --key-uuid ***KeyId***
  : Specifies an MEK key ID that is created by using the **mmkeyserv** command and whose key is used as an MEK. For more information about the **mmkeyserv** command, see IBM Storage Scale documentation.

  --confirm
  : Confirmation by the user to enroll all recovery groups of the node class of the specified recovery group.

The **mmvdisk sed enroll** command completes the following tasks:

- Stores the new RKM ID and the new MEK in the sedKeyId config variable.
- Updates the MEK key from default MSID to the specified new key.
- Enables the drives to get locked when the drives are power recycled.

An error message is displayed, if all the SEDs are not enrolled. If the command fails, it can be rerun to configure all the SEDs by using the same key.

rekey

Uses a new MEK from the RKM server and configures all the SEDs to use a new key as MEK.

--rkmid ***RKMiD***
Specifies a new RKM ID.

--key-uuid ***KeyId***
Updates an MEK key from an old MEK to the specified new key for all SEDs of a recovery group.

--confirm
Confirmation by the user to rekey all recovery groups of the node class of the specified recovery group.

The **mmvdisk sed rekey** command completes the following tasks:

- Updates sedKeyId config variable with the new RKM ID and the new MEK.
- Updates the MEK key from an old MEK to a new key specified for all SEDs of a recovery group.

An error message is displayed, if the command did not run successfully to rekey all the SEDs. If the command fails, it can be rerun to rekey all the SEDs by using the same new key. It required that all the drives are enrolled before running rekey.

list

Displays the SED configuration status of the SEDs of recovery groups. With the pdisk option, the SED configuration status for the given pdisk is displayed. The pdisk can be in states such as Enrolled with sedKeyId, Unenrolled, or info unavailable. It also displays whether the pdisk is locked or unlocked.

--all
Selects all recovery groups.

--recovery-group ***RgName*[,*RgName, ...*]**
Specifies recovery group names.

--pdisk
Specifies the pdisk name of the specified recovery group name.

--pdisk-path ***Pdisk-path***
Specifies the pdisk full path in the //<server_name>/dev/<drive_name> format.

-Y
Displays an output in the machine-readable format.

verify

Verifies whether the drives of a recovery group are SEDs. With the pdisk option, the SED support of the specified pdisk is displayed. The option verifies whether the drive is SED or not.

--all
Selects all recovery groups.

--recovery-group ***RgName*[,*RgName, ...*]**
Specifies recovery group names.

--pdisk
Specifies the pdisk name of the specified recovery group name.

--pdisk-path ***Pdisk-path***
Specifies the pdisk full path in the //<server_name>/dev/<drive_name> format.

-Y
Displays an output in the machine-readable format.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have the root authority to run the **mmvdisk sed** command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Example

1. Enroll a recovery group.

```
# mmvdisk sed enroll --recovery-group rg1_3500_P12N --rkmid rkm_sedKeyId --key-uuid
KEY-86a24d4-13894496-36b6-4688-b638-bfb2698bde39 --confirm
```

A sample output is as follows:

```
mmvdisk: Enrolling disks in recoverygroup rg1_3500_P12N  with new key from default MSID
mmvdisk: Verifying the disks of RG rg1_3500_P12N for SED support.
mmvdisk: Successfully enrolled e1s01 with sedKeyId
mmvdisk: Successfully enrolled e1s02 with sedKeyId
mmvdisk: Successfully enrolled e1s03 with sedKeyId
mmvdisk: Successfully enrolled e1s04 with sedKeyId
mmvdisk: Successfully enrolled e1s05 with sedKeyId
mmvdisk: Successfully enrolled e1s06 with sedKeyId
mmvdisk: Successfully enrolled e1s13 with sedKeyId
mmvdisk: Successfully enrolled e1s14 with sedKeyId
mmvdisk: Successfully enrolled e1s15 with sedKeyId
mmvdisk: Successfully enrolled e1s16 with sedKeyId
mmvdisk: Successfully enrolled e1s17 with sedKeyId
mmvdisk: Successfully enrolled e1s18 with sedKeyId
```

2. Change the MEK on the SEDs to use a new MEK.

```
# mmvdisk sed rekey --recovery-group rg1_3500_P12N --rkmid rkm_sedKeyId --key-uuid
KEY-86a24d4-66b6f796-b178-4778-b45e-2745765d6886
```

A sample output is as follows:

```
mmvdisk: Reenrolling disks in recoverygroup rg1_3500_P12N with new key
mmvdisk: Successfully enrolled e1s01 with sedNewKeyId
mmvdisk: Successfully enrolled e1s02 with sedNewKeyId
mmvdisk: Successfully enrolled e1s03 with sedNewKeyId
mmvdisk: Successfully enrolled e1s04 with sedNewKeyId
mmvdisk: Successfully enrolled e1s05 with sedNewKeyId
mmvdisk: Successfully enrolled e1s06 with sedNewKeyId
mmvdisk: Successfully enrolled e1s13 with sedNewKeyId
mmvdisk: Successfully enrolled e1s14 with sedNewKeyId
mmvdisk: Successfully enrolled e1s15 with sedNewKeyId
mmvdisk: Successfully enrolled e1s16 with sedNewKeyId
mmvdisk: Successfully enrolled e1s17 with sedNewKeyId
mmvdisk: Successfully enrolled e1s18 with sedNewKeyId
```

3. Check the status of drives.

```
# mmvdisk sed list --recovery-group BB01L
```

A sample output is as follows:

```
In nodeclass nc2  SED Configured: True
Disk name      Recovery group     EnrolledStatus/LockedStatus
---------      --------------     ---------------------------
e1s001         BB01L              Enrolled with sedKeyId/Unlocked
e1s002         BB01L              Enrolled with sedKeyId/Unlocked
e1s003         BB01L              Enrolled with sedKeyId/Unlocked
e1s004         BB01L              Enrolled with sedKeyId/Unlocked
e1s005         BB01L              Enrolled with sedKeyId/Unlocked
e1s006         BB01L              Enrolled with sedKeyId/Unlocked
e1s013         BB01L              Enrolled with sedKeyId/Unlocked
e1s014         BB01L              Enrolled with sedKeyId/Unlocked
e1s015         BB01L              Enrolled with sedKeyId/Unlocked
e1s016         BB01L              Enrolled with sedKeyId/Unlocked
e1s017         BB01L              Enrolled with sedKeyId/Unlocked
e1s018         BB01L              Enrolled with sedKeyId/Unlocked
e1s025         BB01L              Enrolled with sedKeyId/Unlocked
e1s026         BB01L              Enrolled with sedKeyId/Unlocked
e1s027         BB01L              Enrolled with sedKeyId/Unlocked
e1s028         BB01L              Enrolled with sedKeyId/Unlocked
e1s029         BB01L              Enrolled with sedKeyId/Unlocked
e1s037         BB01L              Enrolled with sedKeyId/Unlocked
e1s038         BB01L              Enrolled with sedKeyId/Unlocked
e1s039         BB01L              Enrolled with sedKeyId/Unlocked
e1s040         BB01L              Enrolled with sedKeyId/Unlocked
e1s042         BB01L              Enrolled with sedKeyId/Unlocked
e1s049         BB01L              Enrolled with sedKeyId/Unlocked
```

4. Verify whether the drives of recovery groups are SEDs.

```
# mmvdisk sed verify --recovery-group BB01L
```

A sample output is as follows:

```
Disk name      Recovery group     SED Drive
---------      --------------     ---------
e1s001         BB01L              Yes
e1s002         BB01L              Yes
e1s003         BB01L              Yes
e1s004         BB01L              Yes
e1s005         BB01L              Yes
e1s006         BB01L              Yes
e1s013         BB01L              Yes
e1s014         BB01L              Yes
e1s015         BB01L              Yes
e1s016         BB01L              Yes
e1s017         BB01L              Yes
e1s018         BB01L              Yes
e1s025         BB01L              Yes
e1s026         BB01L              Yes
e1s027         BB01L              Yes
e1s028         BB01L              Yes
e1s029         BB01L              Yes
e1s037         BB01L              Yes
e1s038         BB01L              Yes
e1s039         BB01L              Yes
e1s040         BB01L              Yes
```

Location

/usr/lpp/mmfs/bin

mmvdisk nvmeof command

Manages **mmvdisk** NVMeOF target for IBM Storage Scale RAID.

Synopsis

```
mmvdisk nvmeof create  --port [N] {--addr IpAddr | --ifname IfDev}
                          [--family {ipv4|ipv6}] [--type {rdma|tcp}]
                          [--svcid ServicePort]
```

```
                            [--ana-groupid N] [--ana-state State]
                            [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof create  --subsys SubsysName
                            [--cntlid-min N] [--cntlid-max N]
                            [--allowed-hosts {any | Node[,Node ...]}]
                            [--device-path Dev[,Dev ...] [--ana-groupid N]]
                            [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof list  [-N Node[,Node ...] | --node-class NcName] [-Y]
```

or

```
mmvdisk nvmeof config  --vblock --ifname IfDev
                         [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof delete  --subsys {SubsysName|all}
                         [--allowed-hosts {all | Node[,Node ...]} |
                          --device-path Dev[,Dev ...] |
                          --namespace N[,N ...]]
                         [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof delete  --port {N|all}
                            [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof change  --port {N|all} --ana-groupid grpid
                         --ana-state anaState
                         [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof change  --port {N|all} --add --ana-groupid grpid
                            [--ana-state anaState]
                            [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof change  --port {N|all} --delete --ana-groupid grpid
                         [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof change  --subsys {SubsysName|all}
                            [--device-path Dev[,Dev ...] |
                             --namespace N[,N ...]] --ana-groupid grpid
                            [-N Node[,Node ...] | --node-class NcName]]
```

or

```
mmvdisk nvmeof bind --port {N|all} --subsys {SubsysName|all}
                            [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof unbind  --port {N|all} --subsys {SubsysName|all}
                         [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof enable  --subsys {SubsysName|all}
                           [--device-path Dev[,Dev ...] |
                           --namespace N[,N ...]]
                           [-N Node[,Node ...] | --node-class NcName]
```

or

```
mmvdisk nvmeof disable --subsys {SubsysName|all}
                         [--device-path Dev[,Dev ...] |
                          --namespace N[,N ...]]
                         [-N Node[,Node ...] | --node-class NcName]]
```

or

```
mmvdisk nvmeof clear [--confirm] [-N Node[,Node ...] | --node-class NcName]
```

Availability

Available on all IBM Storage Scale editions.

Description

Use the **mmvdisk nvmeof** command to manage the NVMeOF target.

Use this command to do the following tasks:

- Create an NVMe target port.

  Use the **mmvdisk nvmeof create --port [N]** command to create an NVMe target port. The value of *N* specifies the port ID, and it must be either 1 or greater. If *N* value is not specified, the **mmvdisk** command selects an available value as a port ID. A port must be associated with an IP address, which can be specified by using the `--addr` *IPAddr* or `--ifname` *IfDev* option. *IfDev* refers to an InfiniBand device name, such as `mlx5_0/1`, or a network device name, such as `ib0`. The IP address is retrieved by the **mmvdisk** based on the specified InfiniBand or network device. The default protocol is IPv4. The `--family` {*ipv4*|*ipv6*} option can be used to specify a different protocol. The default transfer type is RDMA. The `--type` {*rdma*|*tcp*} option can be used to specify a different transfer type for the NVMe target port. By default, the service port number is set to 4420. It can be changed by using the `--svcid` *ServicePort* option. When creating an NVMe target port, an ANA group with an ID of 1 is automatically created, and the group state is set to optimized. Additional ANA groups can be created by using the `--ana-groupid` *N* option, where *N* is the desired group ID. The state of the new ANA group can be set by using the `--ana-state` option.
- Create an NVMe target subsystem.

  Use the **mmvdisk nvmeof create --subsys SubsysName** command to create an NVMe target subsystem, and to add a new device path or allowed hosts to the subsystem. To allow other nodes to discover and connect to the NVMe target subsystem, use the `--allowed-hosts` option. The specifed *any* value of this option allows any node to discover and connect to the NVMe target subsystem. Multiple node names can be specified as a comma-separated list. The node names must belong to the same cluster as the NVMe target subsystem. The `--device-path` option is used to specify the device path for the NVMe target subsystem. Multiple device paths can be specified as a comma-separated list. If a subsystem with the specified name exists, the specified device paths are added to the existing subsystem. The `--ana-groupid` option is a property of the device path. If you want to specify different device paths with different ANA group IDs under the same NVMe target subsystem, issue the **mmvdisk nvmeof create --subsys SubsysName** command multiple times with a different device path and an ANA group ID. The `--cntlid-min` and `--cntlid-max` options are used to set the minimum and maximum values of the controller identifier (CNTLID). The same *SubsysName* must be on each I/O node for NVMe building blocks with twin-tails, such as IBM ESS 3000/3200/3500, native support for multiple paths of NVMe. In this case, the `--cntlid-min` and `--cntlid-max` options must be set to different scopes for each node. Because a client cannot connect to the same *SubsysName* from different hosts with the same `cntlid` at the same time. For IBM Storage Scale ECE, it is recommended to include the

node number in the *SubsysName*, for example, include *{nodeNumber}* variable in the *SubysName*, which ensures that subsystem names are not identical on different nodes.

- List information about the NVMeOF target.

  Use the **mmvdisk nvmeof list** command to list the NVMeOF target information, including the NVMe target ports and subsystems. The `--discover-subsystem` option displays the NVMeOF `discover.conf` entry form that can be used by NVMe hosts to discover and connect to the NVMeOF target.
- Configure a VDisk over the fabric target.
- Delete an NVMe target subsystem.

  Use the **mmvdisk nvmeof delete --subsys** command to delete one or all NVMe target subsystems, and to remove device paths, allowed hosts, or namespaces from a subsystem. If the `--allowed-hosts` option is specified, allowed hosts are removed from the subsystem. If the `--device-path` or `--namespace` option is specified, a specified device path or namespace is removed from the subsystem. To delete an NVMe target subsystem successfully, ensure that the subsystem is not bound to any NVMe target port before deleting it.
- Delete the NVMe target port.
- Configure an ANA group ID or set ANA group state for an NVMe target port.

  Use the **mmvdisk nvmeof change** command to add or delete ANA groups, change ANA group states under NVMe target ports, and change ANA groups under namespaces of NVMe target subsystems.
- Configure an ANA group ID for the NVMe target subsystem.
- Bind or unbind the NVMe target subsystem to the NVMe target port.
  - Use the **mmvdisk nvmeof bind** command to bind one or all NVMe target subsystems to one or all NVMe target ports.
  - Use the **mmvdisk nvmeof unbind** command to unbind one or all NVMe target subsystems from one or all NVMe target ports.
- Enable or disable the NVMe target subsystem.
  - Use the **mmvdisk nvmeof enable** command to enable one or all NVMe target subsytems. If you specify the `--device-path` or `--namespace` option, it enables the specified namespaces under the specified NVMe target subsystem.
  - Use the **mmvdisk nvmeof disable** command to disable one or all NVMe target subsytems. If you specify the `--device-path` or `--namespace` option, it disables the specified namespaces under the specified NVMe target subsystem.
- Clear all NVMe target.

  Use the **mmvdisk nvmeof clear** command to clear the current NVMe target configuration. This command deletes all NVMe target subsystems and ports.

  Attention: After you run this command, data on the NVMe target subsystems is deleted permanently. Back up important data before you use this command.

Parameters

mmvdisk nvmeof create
Creates an NVMe target subsystem and a port.

mmvdisk nvmeof config
Configures a VDisk over a fabric target.

mmvdisk nvmeof list
Lists an NVMe target port and a subsystem.

mmvdisk nvmeof delete
Delete an NVMe target port or a subsystem.

mmvdisk nvmeof change
Adds or deletes an ANA group, changes ANA group state for an NVMe target port. Changes ANA group for namespace of an NVMe target subsystem.

mmvdisk nvmeof bind
Binds an NVMe target subsystem to an NVMe target port.

mmvdisk nvmeof unbind
Unbinds an NVMe target subsystem from an NVMe target port.

mmvdisk nvmeof enable
Enables an NVMe target subsystem.

mmvdisk nvmeof disable
Disables an NVMe target subsystem.

mmvdisk nvmeof clear
Clears current NVMe target configuration.

--node-class NcName
Specifies a single **mmvdisk** node class name to be created, deleted, changed, bound, unbound, enabled, disabled, or cleared.

-N Node[,Node...]
Specifies individual nodes to be created, deleted, changed, bound, unbound, enabled, disabled, or cleared.

--subsys SubsysName
Specifies an NVMe target subsystem name for use with the **mmvdisk nvmeof create, delete, change, bind, unbind, enable, and disable** commands. *SubsysName* supports following variable: {nodeName}, {shortName}, and {nodeNumber}. When above variable occurs in SubsysName, **mmvdisk** replaces it with the actual value on a target node. For example, **--subsys mysubsys.{nodeNumber}** means **--subsys mysubsys.1**, if the node number of the specified node is 1.

--port {N|all}
Specifies an NVMe target port ID for use with the **mmvdisk nvmeof create, delete, and change** commands. *N* must be greater than 0. The option `all` is used to refer to all existing port IDs. When the port ID is not specified by using the **mmvdisk nvmeof create** command, the **mmvdisk** automatically selects a suitable port ID to create the NVMe target port.

--cntlid-min N
Specifies minimum values of the controller identifier (CNTLID) for use with the **mmvdisk nvmeof create** command to create an NVMe target port.

--cntlid-max N
Specifies maximum values of the controller identifier (CNTLID) for use with the **mmvdisk nvmeof create** command to create an NVMe target port.

--allowed-hosts {any | Node[,Node ...]}
Specifies allowed node that can discover and connect for use with the **mmvdisk nvmeof create** command to create an NVMe target subsystem. Multiple nodes can be specified by separating them with commas. The node name must belong to current cluster. The *any* value indicates that an NVMe target subsystem might be discovered and connected by any hosts. By default, the NVMe target subsystem does not allow any host to discover and connect.

--device-path Dev[,Dev ...]
Specifies a device path for use with the **mmvdisk nvmeof create, delete, change, enable and disable** commands. For example, `/dev/nvme0n1`. It supports the `diskExpr` format also. More information, refer the `--disk-list` *DiskExp* option in the "mmvdisk recoverygroup command" on page 357.

--namespace N[,N ...]
Specifies a namespace for use with the **mmvdisk nvmeof delete, change, enable and disable** commands. *N* must be an existing namespace ID under an NVMe targt subsystem. The device path and namespace have a one-to-one correspondence. Only the `--device-path` or `--namespace` option can be used at a time.

--ana-groupid N

--ana-stat State
Specifies an ANA group ID for use with the **mmvdisk nvmeof create and change** commands. An ANA group is a property of the namespace in the NVMe target subsystem, and its state is managed by the NVMe target port. The value of *N* must be an integer and greater than 0, and the default value is 1. The valid values for the `State` option are *optimized*, *nonoptimized*, *inaccessible*, *persistent-loss*, and *change*.

--addr IpAddr
--ifname IfDev
--family {ipv4|ipv6}
Specifies an IP address for use with the **mmvdisk nvmeof create** commands when you create an NVMe target port. The *IpAddr* value can be specified in the IPv4 or IPv6 format. If you specify the IPv6 address format, set the `--family` option to *ipv6*. The IfDev parameter can refer to either an InfiniBand device name (such as mlx5_0/1) or a network device name (such as ib0). The mmvdisk command retrieves the IP address by specifying the *IfDev* parameter. The default value of the `--family` option is ipv4.

--type {rdma|tcp}
Specifies type for use with the **mmvdisk nvmeof create** commands when creating an NVMe target port. The valid values are *rdma* and *tcp*.

--svcid ServicePort
Specifies a service port for use with **mmvdisk nvmeof create** commands when create an NVMe target port. Its default value is 4420.

--vblock
Specifies exporting VDisk flag for use with the **mmvdisk nvmeof config** commands.

--add
Adds an ANA group to an NVMe target port.

--delete
Deletes an ANA group from an NVMe target port.

--confirm
Confirms the NVMe configuration deletion. This option bypasses an interactive prompt that confirms NVMe configuration deletion.

-Y
Displays a colon-delimited raw output.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the **mmvdisk nvmeof** command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Example

- Create a port by using the `nc3500` node class:

```
# mmvdisk nvmeof create --port --ifname ib0 --type rdma --nc nc3500
```

A sample output is as follows:

```
c145f11san08a: mmvdisk: Create Port '1' addr '13.1.145.132' svcid '4420' type 'rdma'
successfully.
```

```
c145f11san08b: mmvdisk: Create Port '1' addr '13.1.145.134' svcid '4420' type 'rdma'
successfully.
```

- Create a subsystem by using the nc3500 node class.

```
mmvdisk nvmeof create --subsys ustore --device-path 78E4001-[7-12,19-24] --cntlid-min 1 --
cntlid-max 1024 --allowed-hosts any -N c145f11san08a
```

A sample output is as follows:

```
c145f11san08a: mmvdisk: Create Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '1' device path '/dev/nvme16n1(/dev/disk/by-id/nvme-
eui.35344b304ea00582002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '2' device path '/dev/nvme23n1(/dev/disk/by-id/nvme-
eui.35344b304ea00333002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '3' device path '/dev/nvme17n1(/dev/disk/by-id/nvme-
eui.35344b304ea00482002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '4' device path '/dev/nvme24n1(/dev/disk/by-id/nvme-
eui.35344b304ea00579002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '5' device path '/dev/nvme18n1(/dev/disk/by-id/nvme-
eui.35344b304ea00477002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '6' device path '/dev/nvme25n1(/dev/disk/by-id/nvme-
eui.35344b304ea00483002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '7' device path '/dev/nvme8n1(/dev/disk/by-id/nvme-
eui.35344b304ea00586002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '8' device path '/dev/nvme2n1(/dev/disk/by-id/nvme-
eui.35344b304ea00661002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '9' device path '/dev/nvme7n1(/dev/disk/by-id/nvme-
eui.35344b304ea00688002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '10' device path '/dev/nvme1n1(/dev/disk/by-id/nvme-
eui.35344b304ea00414002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '11' device path '/dev/nvme6n1(/dev/disk/by-id/nvme-
eui.35344b304ea00511002538450000004)' for Subsys 'ustore'.
c145f11san08a: mmvdisk: Create namespace '12' device path '/dev/nvme0n1(/dev/disk/by-id/nvme-
eui.35344b304ea00503002538450000004)' for Subsys 'ustore'.
```

- Bind all subsystems to all ports by using the nc3500 node class:

```
/usr/lpp/mmfs/bin/mmvdisk nvmeof bind --port all --subsys all --node-class nc3500
```

A sample output is as follows:

```
c145f11san08a: mmvdisk: Bind Subsys 'ustore' to Port '1'.
c145f11san08b: mmvdisk: Bind Subsys 'ustore' to Port '1'.
```

- List all NVMe target configuration.

```
mmvdisk nvmeof list --nc nc3500
```

A sample output is as follows:

```
 node
number  server         port   type    addr          svcid   subsystems
------  -------------  -----  ------  ------------  ------  ------------
     1  c145f11san08b  1      rdma    13.1.145.134  4420    ustore
     2  c145f11san08a  1      rdma    13.1.145.132  4420    ustore

 node
number  server         subsystem     NS      path           state
------  -------------  ------------  ------  -------------  --------
     1  c145f11san08b  ustore        1     /dev/nvme16n1  Enabled
                                     2     /dev/nvme23n1  Enabled
                                     3     /dev/nvme17n1  Enabled
                                     4     /dev/nvme24n1  Enabled
                                     5     /dev/nvme18n1  Enabled
                                     6     /dev/nvme25n1  Enabled
                                     7     /dev/nvme8n1   Enabled
                                     8     /dev/nvme2n1   Enabled
                                     9     /dev/nvme7n1   Enabled
                                     10    /dev/nvme1n1   Enabled
                                     11    /dev/nvme6n1   Enabled
                                     12    /dev/nvme0n2   Enabled
     2  c145f11san08a                1     /dev/nvme16n1  Enabled
                                     2     /dev/nvme23n1  Enabled
                                     3     /dev/nvme17n1  Enabled
```

```
                                    4     /dev/nvme24n1  Enabled
                                    5     /dev/nvme18n1  Enabled
                                    6     /dev/nvme25n1  Enabled
                                    7     /dev/nvme8n1   Enabled
                                    8     /dev/nvme2n1   Enabled
                                    9     /dev/nvme7n1   Enabled
                                    10    /dev/nvme1n1   Enabled
                                    11    /dev/nvme6n1   Enabled
                                    12    /dev/nvme0n1   Enabled
```

- Clear all NVMe target configuration.

```
mmvdisk nvmeof clear --nc nc3500
```

A sample output is as follows:

```
mmvdisk: This command will clear all nvmet configuration.

mmvdisk: Do you wish to continue (yes or no)? yes
```

See also

- “mmvdisk command” on page 341
- “mmvdisk server command” on page 350
- “mmvdisk recoverygroup command” on page 357
- “mmvdisk filesystem command” on page 373
- “mmvdisk vdiskset command” on page 380
- “mmvdisk vdisk command” on page 386
- “mmvdisk nodeclass command” on page 346

Location

/usr/lpp/mmfs/bin

Appendix C. IBM Storage Scale RAID scripts

This section includes descriptions of the IBM Storage Scale RAID scripts.

Descriptions of these scripts follow:

For information about IBM Storage Scale RAID commands, see Appendix B, "IBM Storage Scale RAID commands," on page 263.

For information about other IBM Storage Scale commands, see the *IBM Storage Scale: Command and Programming Reference*.

chdrawer script

A service aid for replacing an ESS enclosure drawer.

Synopsis

```
chdrawer  EnclosureSerialNumber DrawerNumber
     {--release  | --replace } [--dry-run]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `chdrawer` script acts as a service aid for replacing an ESS enclosure drawer. For more information, see *Replacing a failed ESS storage drawer: a sample scenario* in *Elastic Storage Server: Problem Determination Guide*.

Parameters

EnclosureSerialNumber
Specifies the enclosure serial number, as displayed by `mmlsenclosure`.

DrawerNumber
Specifies the drawer number to be replaced. Drawers are numbered from top to bottom in an enclosure.

`--release`
Prepares the drawer for disk replacement by suspending all pdisks.

`--replace`
Resumes all the pdisks once the drawer has been replaced (with all the former pdisks in their corresponding slots).

`--dry-run`
Runs the command without actually changing the pdisk states; checks whether there is enough redundancy to safely replace the drawer on a live system.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `chdrawer` script.

The node on which the script is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

See also

See also the following *IBM Storage Scale RAID: Administration* topic:

- "mmchenclosure command" on page 277

Location

`/usr/lpp/mmfs/samples/vdisk`

gnrhealthcheck script

Checks the general health of an ESS configuration.

Synopsis

```
gnrhealthcheck  [--topology] [--enclosure] [--rg] [--pdisk]
                 [--perf-dd] [--ipr] [--nvme-ctrl] [--ssd-endurance-percentage] [--local]
```

Availability

Available on all IBM Storage Scale editions.

Description

The `gnrhealthcheck` script checks the general health of an ESS configuration.

Parameters

`--topology`
Checks the operating system topology. Runs `mmgetpdisktopology` and `topsummary` to look for cabling and path issues.

`--enclosure`
Checks enclosures. Runs `mmlsenclosure` to look for failures.

`--rg`
Checks recovery groups. Runs `mmlsrecoverygroup` to check whether all recovery groups are active and whether the active server is the primary server. Also checks for any recovery groups that need service.

`--pdisk`
Checks pdisks. Runs `mmlspdisk` to check that each pdisk has two paths.

--perf-dd
Checks basic performance of disks. Runs a dd read to each potential IBM Storage Scale RAID disk drive for a GB and reports basic performance statistics. Reads are done six disks at a time. These statistics will only be meaningful if run on an idle system. Available on Linux only.

--ipr
Checks IBM Power RAID Array status. Runs `iprconfig` to check if the local RAID adapter is running "Optimized" or "Degraded". The ESS NVR pdisks are created on a RAID 10 array on this adapter. If one of the drives has failed, it will affect performance and should be replaced.

--nvme-ctrl
Checks NVMe controllers. Runs **mmlsnvmestatus** to find the status of the NVMe controllers.

--ssd-endurance-percentage
Runs **mmlspdisk** to check for SSDs that exceed 90% endurance.

--local
Runs tests only on the invoking node.

The default is to check everything *except* `--perf-dd` arguments on all NSD server nodes.

Exit status

0
No problems were found.

1
Problems were found and information was displayed.

Note: The default is to display to standard output. There could be a large amount of data, so it is recommended that you pipe the output to a file.

Security

You must have root authority to run the `gnrhealthcheck` script.

The node on which the script is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more details, see the following *IBM Storage Scale RAID: Administration* topic: “Requirements for administering IBM Storage Scale RAID” on page 17.

Examples

1. In this example, all checks are successful.

   To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   The system displays information similar to this:

   ```
   ################################################################################
   # Beginning topology checks.
   ################################################################################
   Topology checks successful.

   ################################################################################
   # Beginning enclosure checks.
   ################################################################################
   Enclosure checks successful.

   ################################################################################
   # Beginning recovery group checks.
   ################################################################################
   Recovery group checks successful.

   ################################################################################
   # Beginning pdisk checks.
   ```

```
##########################################################################
Pdisk group checks successful.

##########################################################################
# Beginning IBM Power RAID checks.
##########################################################################
IBM Power RAID checks successful.

##########################################################################
# Beginning the NVMe Controller checks.
##########################################################################
The NVMe Controller checks are successful.

##########################################################################
# Beginning SSD endurance checks
##########################################################################
The SSD endurance checks are successful.
```

2. In this example, several issues need to be investigated.

   To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   The system displays information similar to this:

```
##################################################################
# Beginning topology checks.
##################################################################
Found topology problems on node c45f01n01-ib0.gpfs.net

DCS3700 enclosures found: 0123456789AB SV11812206 SV12616296 SV13306129
Enclosure 0123456789AB (number 1):
Enclosure 0123456789AB ESM A sg244[0379][scsi8 port 4] ESM B sg4[0379][scsi7 port 4]
Enclosure 0123456789AB Drawer 1 ESM sg244 12 disks diskset "19968" ESM sg4 12 disks diskset "19968"
Enclosure 0123456789AB Drawer 2 ESM sg244 12 disks diskset "11294" ESM sg4 12 disks diskset "11294"
Enclosure 0123456789AB Drawer 3 ESM sg244 12 disks diskset "60155" ESM sg4 12 disks diskset "60155"
Enclosure 0123456789AB Drawer 4 ESM sg244 12 disks diskset "03345" ESM sg4 12 disks diskset "03345"
Enclosure 0123456789AB Drawer 5 ESM sg244 11 disks diskset "33625" ESM sg4 11 disks diskset "33625"
Enclosure 0123456789AB sees 59 disks

Enclosure SV12616296 (number 2):
Enclosure SV12616296 ESM A sg63[0379][scsi7 port 3] ESM B sg3[0379][scsi9 port 4]
Enclosure SV12616296 Drawer 1 ESM sg63 11 disks diskset "51519" ESM sg3 11 disks diskset "51519"
Enclosure SV12616296 Drawer 2 ESM sg63 12 disks diskset "36246" ESM sg3 12 disks diskset "36246"
Enclosure SV12616296 Drawer 3 ESM sg63 12 disks diskset "53750" ESM sg3 12 disks diskset "53750"
Enclosure SV12616296 Drawer 4 ESM sg63 12 disks diskset "07471" ESM sg3 12 disks diskset "07471"
Enclosure SV12616296 Drawer 5 ESM sg63 11 disks diskset "16033" ESM sg3 11 disks diskset "16033"
Enclosure SV12616296 sees 58 disks

Enclosure SV11812206 (number 3):
Enclosure SV11812206 ESM A sg66[0379][scsi9 port 3] ESM B sg6[0379][scsi8 port 3]
Enclosure SV11812206 Drawer 1 ESM sg66 11 disks diskset "23334" ESM sg6 11 disks diskset "23334"
Enclosure SV11812206 Drawer 2 ESM sg66 12 disks diskset "16332" ESM sg6 12 disks diskset "16332"
Enclosure SV11812206 Drawer 3 ESM sg66 12 disks diskset "52806" ESM sg6 12 disks diskset "52806"
Enclosure SV11812206 Drawer 4 ESM sg66 12 disks diskset "28492" ESM sg6 12 disks diskset "28492"
Enclosure SV11812206 Drawer 5 ESM sg66 11 disks diskset "24964" ESM sg6 11 disks diskset "24964"
Enclosure SV11812206 sees 58 disks

Enclosure SV13306129 (number 4):
Enclosure SV13306129 ESM A sg64[0379][scsi8 port 2] ESM B sg353[0379][scsi7 port 2]
Enclosure SV13306129 Drawer 1 ESM sg64 11 disks diskset "47887" ESM sg353 11 disks diskset "47887"
Enclosure SV13306129 Drawer 2 ESM sg64 12 disks diskset "53906" ESM sg353 12 disks diskset "53906"
Enclosure SV13306129 Drawer 3 ESM sg64 12 disks diskset "35322" ESM sg353 12 disks diskset "35322"
Enclosure SV13306129 Drawer 4 ESM sg64 12 disks diskset "37055" ESM sg353 12 disks diskset "37055"
Enclosure SV13306129 Drawer 5 ESM sg64 11 disks diskset "16025" ESM sg353 11 disks diskset "16025"
Enclosure SV13306129 sees 58 disks

DCS3700 configuration: 4 enclosures, 1 SSD, 7 empty slots, 233 disks total
Location 0123456789AB-5-12 appears empty but should have an SSD
Location SV12616296-1-3 appears empty but should have an SSD
Location SV12616296-5-12 appears empty but should have an SSD
Location SV11812206-1-3 appears empty but should have an SSD
Location SV11812206-5-12 appears empty but should have an SSD

scsi7[07.00.00.00] 0000:11:00.0 [P2 SV13306129 ESM B (sg353)] [P3 SV12616296 ESM A (sg63)] [P4 0123456789AB ESM B (sg4)]
scsi8[07.00.00.00] 0000:8b:00.0 [P2 SV13306129 ESM A (sg64)] [P3 SV11812206 ESM B (sg6)] [P4 0123456789AB ESM A (sg244)]
scsi9[07.00.00.00] 0000:90:00.0 [P3 SV11812206 ESM A (sg66)] [P4 SV12616296 ESM B (sg3)]

##################################################################
# Beginning enclosure checks.
##################################################################
Enclosure checks successful.

##################################################################
# Beginning recovery group checks.
##################################################################
Found recovery group BB1RGR, primary server is not the active server.

##################################################################
# Beginning pdisk checks.
##################################################################
Found recovery group BB1RGL pdisk e4d5s06 has 0 paths.

##################################################################
```

```
# Beginning IBM Power RAID checks.
##################################################################
IBM Power RAID Array is running in degraded mode.

Name   PCI/SCSI Location          Description               Status
------ -------------------------  ------------------------- -----------------
       0007:90:00.0/0:            PCI-E SAS RAID Adapter    Operational
       0007:90:00.0/0:0:1:0       Advanced Function Disk    Failed
       0007:90:00.0/0:0:2:0       Advanced Function Disk    Active sda
       0007:90:00.0/0:2:0:0       RAID 10  Disk Array       Degraded
       0007:90:00.0/0:0:0:0       RAID 10  Array Member     Active
       0007:90:00.0/0:0:3:0       RAID 10  Array Member     Failed
       0007:90:00.0/0:0:4:0       Enclosure                 Active
       0007:90:00.0/0:0:6:0       Enclosure                 Active
       0007:90:00.0/0:0:7:0       Enclosure                 Active

##################################################################
# Beginning the NVMe Controller checks.
##################################################################
The NVMe Controller checks are successful.

##################################################################
# Beginning SSD endurance checks
##################################################################
The SSD endurance checks are successful.
```

See also

See also the following *Elastic Storage Server: Problem Determination Guide* topic:

- *Checking the health of an ESS configuration: a sample scenario*

Location

`/usr/lpp/mmfs/samples/vdisk`

mkrginput script

Generates stanza files for recovery group creation from IBM Storage Scale RAID topology files.

Synopsis

```
mkrginput  TopologyFile1 TopologyFile2 [-s | -m ] [ --match percent ]
```

Availability

Available on all IBM Storage Scale editions.

Description

The **mkrginput** script generates the stanza files needed for the creation of IBM Storage Scale RAID recovery groups by the **mmcrrecoverygroup** command. It takes as input the topology files generated by the **mmgetpdisktopology** command on the two servers that will share the recovery groups, and generates as output two stanza files for use by the **mmcrrecoverygroup** command.

The **mkrginput** script tries to verify that the two topology files are from different servers that share the same set of disk enclosures, and will exit with an error if this is not the case.

Despite this checking, it is imperative that the topology files first be examined with the **topsummary** command to ensure that both of them reflect the intended IBM Storage Scale RAID disk topology. If there are any disk or cabling errors, they should be corrected, and the topology files should be reacquired with **mmgetpdisktopology** and verified with **topsummary** before they are used with **mkrginput** to generate the stanza files.

After the topologies have been verified to be correct, they can be supplied as command arguments to **mkrginput**. The **mkrginput** command will then generate a left and right stanza file, each named with the serial number of the first disk enclosure. The left and right recovery groups can then be created by supplying the respective stanza files and servers in two separate **mmcrrecoverygroup** commands.

To create recovery group stanza files, follow these steps:

1. On the first server, `server1`, run:

```
mmgetpdisktopology > server1.top
```

2. On the second server, `server2`, run:

```
mmgetpdisktopology > server2.top
```

3. To verify the correctness of `server1`'s topology, run:

```
topsummary server1.top
```

4. To verify the correctness of `server2`'s topology, run:

```
topsummary server2.top
```

5. Repeat steps 1 - 4, correcting any disk subsystem problems until both topologies reflect the intended configuration.
6. Run:

```
mkrginput server1.top server2.top
GNR server disk topology: ESS GS6 HDD
GNR recovery group layout: GS GSS/ESS RG
Number of recovery groups: 2
Created stanza file: G46503NL.stanza
Created stanza file: G46503NR.stanza
Stanza file contains '#%vdisk:' lines with log vdisk attributes.
```

   The **mkrginput** command identifies the server topology that was detected in the server topologies and the recovery group layout that is used for the detected topology. It prescribes that two recovery groups be created, and gives the names of the stanza files that have been created for them. The stanza files will have commented out `%vdisk` stanzas for use after the recovery groups have been created, with suggested values for all the required log vdisks.

   At this point look in the current directory for two files, one named *serialnumber*`L.stanza` and one named *serialnumber*`R.stanza`. Suppose the first enclosure in the topologies has the serial number `G46503N`. The **mkrginput** command will have created two files: `G46503NL.stanza` and `G46503NR.stanza`. Suppose the left recovery group will be named `RGL` and the right recovery group will be named `RGR`.
7. To create the left recovery group, `RGL`, run:

```
mmcrrecoverygroup RGL -F G46503NL.stanza --servers server1,server2
```

8. To create the right recovery group, `RGR`, run:

```
mmcrrecoverygroup RGR -F G46503NR.stanza --servers server2,server1
```

It is imperative that the disk topology is not changed between the capture of the topology files with **mmgetpdisktopology** and the creation of the two recovery groups with **mmcrrecoverygroup**. No server reboots or disk enclosure changes should be performed after the topologies have been verified as correct with **topsummary** until after recovery group creation.

After the recovery groups have been created, the commented-out `%vdisk` stanzas can be extracted from the recovery group input file and edited to include the recovery group names:

```
grep %vdisk G46503NL.stanza
#%vdisk: vdiskName={rg}LOGTIP rg={rg} da=NVR diskUsage=vdiskLogTip raidCode=2WayReplication blocksize=2m size=48m
#%vdisk: vdiskName={rg}LOGTIPBACKUP rg={rg} da=SSD diskUsage=vdiskLogTipBackup raidCode=Unreplicated blocksize=2m size=48m
#%vdisk: vdiskName={rg}LOGHOME rg={rg} da=DA1 diskUsage=vdiskLog raidCode=4WayReplication blocksize=2m size=60g
                  longTermEventLogSize=4m shortTermEventLogSize=4m fastWriteLogPct=90
```

The string `{rg}` should be replaced with the name of the recovery group that was created using this stanza file. The comment symbol # should be removed, and the resulting vdisk stanza file may be supplied to the **mmcrvdisk** command to create the required log vdisks for the recovery group. The process should be repeated for the second recovery group. See the "mmcrvdisk command" on page 291 command for more information.

Parameters

TopologyFile1
Specifies the name of the topology file from the first server.

TopologyFile2
Specifies the name of the topology file from the second server.

-s
Creates a single large declustered array. It is the default.

-m
This is a deprecated legacy option to create multiple smaller declustered arrays. It should only be used to maintain compatibility with certain GSS and ESS server topologies where there are preexisting recovery groups with multiple declustered arrays. The default is `-s`.

--match percent
Specifies the minimum percentage match of the number of disks and their locations in the server topologies, as reported by the **topsummary** command. The default is 100, since it is not recommended to create recovery groups from imperfect server topologies. Valid values are 75 - 100, inclusive.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the **mkrginput** script.

The node on which the script is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- "mmcrrecoverygroup command" on page 288
- "mmcrvdisk command" on page 291
- "mmgetpdisktopology command" on page 305
- "topsummary script" on page 415

Location

`/usr/lpp/mmfs/samples/vdisk`

topselect script

Selects enclosures and disks from an IBM Storage Scale RAID topology file.

Synopsis

```
topselect { -l | [ -d DiskEnclosure[,DiskEnclosure] ]
[ -a Adapter[,Adapter] ]  [ -p EnclosurePort[,EnclosurePort] ]
[-n] } TopologyFile
```

Availability

Available on all IBM Storage Scale editions.

Description

The **topselect** script provides a simple interface for examining the contents of a topology file produced by the **mmgetpdisktopology** command. It produces two types of output. One output type lists the supported enclosures that are found in the topology file and the other output type lists selected subsets of the disk block devices in the topology.

Together with the **topsummary** script, this script can be used to read information out of a IBM Storage Scale RAID topology file. It is sometimes useful to know which disk devices are connected to what hardware: for example, when looking to see if a pattern of disk faults might be isolated to a certain adapter.

You can list the enclosures within a topology file using the `-l` option:

```
topselect -l server1.top
Enclosure GSS SV34607290 found
Adapter scsi5 (naa.500605B00687D3A0) connects to SV34607290 ESM(s) B
Adapter scsi7 (naa.500605B006940730) connects to SV34607290 ESM(s) A
Enclosure GSS SV34607449 found
Adapter scsi3 (naa.500605B006BB9AD0) connects to SV34607449 ESM(s) B
Adapter scsi5 (naa.500605B00687D3A0) connects to SV34607449 ESM(s) A
```

Two disk enclosures are represented in the topology file `server1.top`. It shows that there are two connections to the enclosure with serial number SV34607290: one over adapter `scsi5` to ESM B and the other over adapter `scsi7` to ESM A. In parentheses after each adapter is the unique WWID for that adapter; this information can be useful for diagnostic purposes, together with the physical location codes and firmware levels displayed by the **topsummary** script.

The default behavior of the **topselect** script is to list disk block devices within the topology file. When used with no options, all the disk block devices contained within all disk enclosures are listed. The available IBM Storage Scale RAID log tip NVRAM devices can be included with the `-n` option. The listing of disk block devices can be narrowed down by using a combination of the options `-d` for disk enclosures, `-a` for adapters, and `-p` for enclosure ports (ESMs).

The information listed for disk block devices follows:

```
Enclosure:  Enclosure serial number
ESM port:   A or B
Adapter:    SCSI hostbus of the adapter
SES device: /dev name of the ESM expander controlling the disk
Device:     /dev name of the disk block device
Type:       SSD, HDD, or NVR
WWID:       Unique WWID of the disk
Location:   Disk location within the enclosure
```

For example, the /dev/sdic disk block device might appear in the output of **topselect** as:

```
SX33100383 B scsi4 sg249 sdic HDD 5000C5006C2C5837 21
```

Parameters

-l
Lists the enclosure connections present within the topology.

-d *DiskEnclosure*[,*DiskEnclosure*]
Lists the disk block devices in the specified disk enclosures.

-a *Adapter*[,*Adapter*]
Lists the disk block devices accessed over the specified adapters.

-p *EnclosurePort*[,*EnclosurePort*]
Lists the disk block devices accessed over the specified ESMs.

-n
Includes the available NVRAM block devices present in the topology.

TopologyFile
Specifies the name of the topology file from which to select.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **topselect** script.

The node on which the script is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Examples

In the following examples, the underscores in the FC5887 disk location codes are used by the **topselect** script in place of literal space characters in the actual location code.

1. To select the disk block devices on ESM B of enclosure G46600Y in the server1.top topology file, run:

   ```
   topselect -d G46600Y -p B server1.top
   ```

   The system displays output similar to this:

   ```
   > topselect -d G46600Y -p B server1.top

   G46600Y B scsi6 sg232 sdgp HDD 5000C50067BC9C63 _P1-D5_____2SS6
   G46600Y B scsi6 sg232 sdhd HDD 5000C50067BC9CA7 _P1-D19_____2SS6
   G46600Y B scsi6 sg232 sdgo HDD 5000C50067E9D6CB _P1-D4_____2SS6
   G46600Y B scsi6 sg232 sdgq HDD 5000C50067E9D7A3 _P1-D6_____2SS6
   G46600Y B scsi6 sg232 sdgt HDD 5000C50067E9DA17 _P1-D9_____2SS6
   G46600Y B scsi6 sg232 sdhf HDD 5000C500686881FF _P1-D21_____2SS6
   G46600Y B scsi6 sg232 sdgw HDD 5000C50068688213 _P1-D12_____2SS6
   G46600Y B scsi6 sg232 sdgv HDD 5000C5006868827F _P1-D11_____2SS6
   G46600Y B scsi6 sg232 sdgu HDD 5000C500686884A3 _P1-D10_____2SS6
   G46600Y B scsi6 sg232 sdhb HDD 5000C50068688673 _P1-D17_____2SS6
   ```

```
G46600Y B scsi6 sg232 sdgm HDD 5000C50068688793 _P1-D2_____2SS6
G46600Y B scsi6 sg232 sdgz HDD 5000C500686889E7 _P1-D15____2SS6
G46600Y B scsi6 sg232 sdhi HDD 5000C5006868947F _P1-D24____2SS6
G46600Y B scsi6 sg232 sdgx HDD 5000C5006868AA37 _P1-D13____2SS6
G46600Y B scsi6 sg232 sdgl HDD 5000C500686B84F7 _P1-D1_____2SS6
G46600Y B scsi6 sg232 sdgn HDD 5000C500686BAB57 _P1-D3_____2SS6
G46600Y B scsi6 sg232 sdha HDD 5000C500686BB0F3 _P1-D16____2SS6
G46600Y B scsi6 sg232 sdhc HDD 5000C500686BD21B _P1-D18____2SS6
G46600Y B scsi6 sg232 sdgy HDD 5000C500686BD333 _P1-D14____2SS6
G46600Y B scsi6 sg232 sdhg HDD 5000C500686BD387 _P1-D22____2SS6
G46600Y B scsi6 sg232 sdhe HDD 5000C500686BF457 _P1-D20____2SS6
G46600Y B scsi6 sg232 sdgs HDD 5000C500686BFF37 _P1-D8_____2SS6
G46600Y B scsi6 sg232 sdhh HDD 5000C5006B9F0D97 _P1-D23____2SS6
G46600Y B scsi6 sg232 sdgr HDD 5000C5006BA184EB _P1-D7_____2SS6
```

2. To list all the disk block devices in the `server1.top` topology file and find only the disk with WWID `5000c500686bd333`, use **`topselect`** in a pipeline with `grep`, run:

```
topselect server1.top | grep -i 5000c500686bd333
```

   The system displays output similar to this:

```
> topselect server1.top | grep -i 5000c500686bd333

G46600Y B scsi6 sg232 sdgy HDD 5000C500686BD333 _P1-D14____2SS6
G46600Y A scsi2 sg32 sdo HDD 5000C500686BD333 _P1-D14____2SS6
```

   This shows the two disk block device paths for the disk in slot 14 (D14).

3. To list the disk device paths found over HBA `scsi9` and ESM A for the attached enclosure, run:

```
topselect -a scsi9 -p A server1.top
```

   The system displays output similar to this:

```
> topselect -a scsi9 -p A server1.top

SV21313978 A scsi9 sg365 sdpb SSD 500051610005F8A8 5-12
SV21313978 A scsi9 sg365 sdmw SSD 500051610005F8EC 1-3
SV21313978 A scsi9 sg365 sdod HDD 5000C50034239FC7 3-12
SV21313978 A scsi9 sg365 sdnz HDD 5000CCA01B119598 3-8
.
.
.
SV21313978 A scsi9 sg365 sdni HDD 5000CCA01C515230 2-3
SV21313978 A scsi9 sg365 sdne HDD 5000CCA01C515294 1-11
.
.
.
```

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mmgetpdisktopology command” on page 305
- “mmlspdisk command” on page 324
- “topsummary script” on page 415.

Location

`/usr/lpp/mmfs/samples/vdisk`

topsummary script

Summarizes the contents of an IBM Storage Scale RAID disk subsystem topology file.

Synopsis

```
topsummary  TopologyFile
```

Availability

Available on all IBM Storage Scale editions.

Description

The **topsummary** script examines the topology file produced by the **mmgetpdisktopology** command and prints a concise summary of the disk enclosures and their cabling. Any discrepancies from what is expected will be noted. This is useful in verifying that IBM Storage Scale RAID servers are cabled correctly to the disk enclosures they manage. Discrepancies should be corrected and verified before creating IBM Storage Scale RAID recovery groups.

The typical scenario for using the **topsummary** script will be in the preparation and verification of a IBM Storage Scale RAID server building block. The servers are cabled to the disk enclosures according to specification. The **mmgetpdisktopology** command will be run to capture a topology file for each server. Then the **topsummary** script will be used on each topology file to verify that the servers are correctly cabled and that the expected disks are present. If no discrepancies are noted in either of the server disk topologies, the next step is to proceed to recovery group creation using **mkrginput** and **mmcrrecoverygroup**.

If discrepancies are found, **topsummary** describes the problem with information that should be useful in correcting it. Among the problems that **topsummary** detects are:

- Incorrect cabling
- Missing enclosures
- Missing disks or paths to disks.

If no errors are indicated, the topology matches a supported ESS / IBM Storage Scale RAID configuration. It is remotely possible, though, that this is still an unintended topology. For example, if enclosures 3 and 4 in an intended four-enclosure topology are missing, the remaining two enclosures could look exactly like a two-enclosure topology. It is imperative to know the intended structure of the IBM Storage Scale RAID configuration.

Enclosures are identified by cabling order and serial number. Disks are identified in sets based on the enclosure or drawer in which they reside. The "diskset" is indicated by a 5-digit checksum on the WWIDs of the disks in the set. Adapters are identified by their bus number and address or slot location. The firmware levels of adapters and enclosure ESMs are also provided for easy reference. The total number of enclosures and disks is indicated. The source of the adapter firmware levels found in the topology file is indicated by "[cli]" if the topology was acquired using the adapter CLI, or by "[sys]" if only sysfs information was available.

The total number of enclosures and disks is indicated. The name of the matching topology is provided with a "match" metric that indicates the degree to which the disk locations and contents match the named topology. A match of "100/100" means that all the expected disks were found in the expected locations.

Parameters

TopologyFile
Specifies the name of the topology file to summarize.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **topsummary** script.

The node on which the script is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *IBM Storage Scale RAID: Administration* topic: "Requirements for administering IBM Storage Scale RAID" on page 17.

Example

The following command example shows how to acquire and summarize the topology file on an IBM Storage Scale RAID server:

```
mmgetpdisktopology > server1.top

topsummary server1.top
GNR server: name server1.ibm.net arch ppc64 model 8284-22A serial 022168C6V
GNR enclosures found: G46503N G465013 G46502A G46600J G46600G G46600Y
Enclosure G46503N (IBM 5887, number 1):
Enclosure G46503N ESM A sg307[60RG][scsi7 port 4] ESM B sg207[60RG][scsi5 port 4]
Enclosure G46503N ESM sg307 24 disks diskset "02977" ESM sg207 24 disks diskset "02977"
Enclosure G46503N sees 24 disks (2 SSDs, 22 HDDs)

Enclosure G465013 (IBM 5887, number undetermined):
Enclosure G465013 ESM B sg107[60RG][scsi3 port 4] ESM A not found
Enclosure G465013 ESM sg107 24 disks diskset "28956"
Enclosure G465013 sees 24 disks (0 SSDs, 24 HDDs)

Enclosure G46502A (IBM 5887, number 3):
Enclosure G46502A ESM A sg82[60RG][scsi3 port 3] ESM B sg282[60QG][scsi7 port 3]
Enclosure G46502A ESM sg82 24 disks diskset "41537" ESM sg282 24 disks diskset "41537"
Enclosure G46502A sees 24 disks (0 SSDs, 24 HDDs)

Enclosure G46600J (IBM 5887, number 4):
Enclosure G46600J ESM A sg257[60RG][scsi6 port 2] ESM B sg157[60RG][scsi4 port 2]
Enclosure G46600J ESM sg257 24 disks diskset "33442" ESM sg157 24 disks diskset "33442"
Enclosure G46600J sees 24 disks (0 SSDs, 24 HDDs)

Enclosure G46600G (IBM 5887, number 5):
Enclosure G46600G ESM A sg132[60RG][scsi4 port 1] ESM B sg57[60RG][scsi0 port 2]
Enclosure G46600G ESM sg132 24 disks diskset "20820" ESM sg57 24 disks diskset "20820"
Enclosure G46600G sees 24 disks (0 SSDs, 24 HDDs)

Enclosure G46600Y (IBM 5887, number 6):
Enclosure G46600Y ESM A sg32[60RG][scsi0 port 1] ESM B sg232[60RG][scsi6 port 1]
Enclosure G46600Y ESM sg32 24 disks diskset "34159" ESM sg232 24 disks diskset "34159"
Enclosure G46600Y sees 24 disks (0 SSDs, 24 HDDs)

GNR server disk topology: ESS GS6 HDD (match: 100/100)
GNR configuration: 6 enclosures, 2 SSDs, 0 empty slots, 144 disks total, 6 NVRAM partitions
Unable to determine enclosure order. Check the HBA to enclosure cabling.
Location G465013-1 appears only on the sg107 path
Location G465013-2 appears only on the sg107 path
Location G465013-3 appears only on the sg107 path
Location G465013-4 appears only on the sg107 path
Location G465013-5 appears only on the sg107 path
Location G465013-6 appears only on the sg107 path
Location G465013-7 appears only on the sg107 path
Location G465013-8 appears only on the sg107 path
Location G465013-9 appears only on the sg107 path
Location G465013-10 appears only on the sg107 path
Location G465013-11 appears only on the sg107 path
Location G465013-12 appears only on the sg107 path
Location G465013-13 appears only on the sg107 path
Location G465013-14 appears only on the sg107 path
Location G465013-15 appears only on the sg107 path
Location G465013-16 appears only on the sg107 path
Location G465013-17 appears only on the sg107 path
Location G465013-18 appears only on the sg107 path
Location G465013-19 appears only on the sg107 path
Location G465013-20 appears only on the sg107 path
Location G465013-21 appears only on the sg107 path
Location G465013-22 appears only on the sg107 path
```

```
Location G465013-23 appears only on the sg107 path
Location G465013-24 appears only on the sg107 path

Slot C2 HBA model LSISAS2308 firmware[cli] 20.00.02.00 bios[cli] 07.37.00.00 uefi[cli] 07.26.01.00
Slot C2 HBA scsi4 U78CB.001.WZS01V4-P1-C2-T1 [P1 G46600G ESM A (sg132)] [P2 G46600J ESM B (sg157)]
Slot C2 HBA scsi5 U78CB.001.WZS01V4-P1-C2-T2 [P4 G46503N ESM B (sg207)]
Slot C3 HBA model LSISAS2308 firmware[cli] 20.00.02.00 bios[cli] 07.37.00.00 uefi[cli] 07.26.01.00
Slot C3 HBA scsi6 U78CB.001.WZS01V4-P1-C3-T1 [P1 G46600Y ESM B (sg232)] [P2 G46600J ESM A (sg257)]
Slot C3 HBA scsi7 U78CB.001.WZS01V4-P1-C3-T2 [P3 G46502A ESM B (sg282)] [P4 G46503N ESM A (sg307)]
Slot C11 HBA model LSISAS2308 firmware[cli] 20.00.02.00 bios[cli] 07.37.00.00 uefi[cli] 07.26.01.00
Slot C11 HBA scsi0 U78CB.001.WZS01V4-P1-C11-T1 [P1 G46600Y ESM A (sg32)] [P2 G46600G ESM B (sg57)]
Slot C11 HBA scsi3 U78CB.001.WZS01V4-P1-C11-T2 [P3 G46502A ESM A (sg82)] [P4 G465013 ESM B (sg107)]
```

This shows an ESS GS6 HDD topology. This is a six-enclosure topology with 2 SSDs and 142 HDDs. For illustration purposes, this topology contains a single cabling error. The error is indicated by the undetermined enclosure order, the message that enclosure G465013 ESM A is not found, and by 24 disks that can only be found over one path. The cabling error can also be seen in the missing P3 connection on the adapter in slot C2. The problem here is the missing ESM A connection from enclosure G465013 to port 3 of the adapter in slot C2. (When an enclosure's ESM A is connected to port 3 of slot C2 and its ESM B is connected to port 4 of slot C11, only then will **topsummary** conclude that it is enclosure number 2 in an ESS GS6 HDD topology.)

A close look at the firmware levels shows all of the adapter firmware levels to have been acquired using the adapter CLI, with base firmware at level `20.00.02.00`, BIOS firmware at level `07.37.00.00`, and UEFI firmware at level `07.26.01.00`. But one of the ESMs, enclosure G46502A ESM B, is at a lower level (60QG) than the others (60RG). This is not necessarily an error, but is worth taking note of, and ESS provides other tools for managing firmware levels.

See also

See also the following *IBM Storage Scale RAID: Administration* topics:

- “mkrginput script” on page 409
- “mmcrrecoverygroup command” on page 288
- “mmgetpdisktopology command” on page 305
- “mmlsenclosure command” on page 314
- “mmlsfirmware command” on page 320
- “topselect script” on page 412.

Location

`/usr/lpp/mmfs/samples/vdisk`

Appendix D. Setting up GNR on the Power 775 Disk Enclosure

The IBM Power 775 Disk Enclosure includes version 3.5 of GPFS Native RAID (GNR). This topic includes sample scenarios for setting up GNR and replacing disks on the Power 775 Disk Enclosure.

Disk replacement recording and reporting

The disk hospital keeps track of disks that require replacement according to the disk replacement policy of the declustered array, and it can be configured to report the need for replacement in a variety of ways. It records and reports the FRU number and physical hardware location of failed disks to help guide service personnel to the correct location with replacement disks.

If the storage JBOD supports multiple disks that are mounted on a removable carrier, such as the Power 775, disk replacement requires the hospital to suspend other disks in the same carrier temporarily. On the Power 775 storage JBOD, the disk carriers are also not removable until GNR actuates a solenoid-controlled latch, in order to guard against human error.

In response to administrative commands, the hospital quiesces the appropriate disk (or multiple disks on a carrier), releases the carrier latch solenoid (if necessary), and turns on identify lights to guide replacement. After one or more disks are replaced and the disk or carrier is re-inserted, the hospital, in response to administrative commands, verifies that the repair has taken place and adds any new disks to the declustered array automatically, which causes GPFS Native RAID to rebalance the tracks and spare space across all of the disks of the declustered array. If service personnel fail to re-insert the disk or carrier within a reasonable period, the hospital declares the disks missing and starts rebuilding the affected data.

Configuring GNR recovery groups: a sample scenario

This topic provides a detailed example of configuring GNR using the JBOD SAS disks on the Power 775 Disk Enclosure.

The example considers one fully populated Power 775 Disk Enclosure cabled to two recovery group servers, and shows how the architecture of the Power 775 Disk Enclosure determines the structure of the recovery groups. Throughout this topic, it may be helpful to have Power 775 Disk Enclosure documentation at hand.

Preparing recovery group servers

Disk enclosure and HBA cabling

The Power 775 Disk Enclosure should be cabled to the intended recovery group servers according to the Power 775 Disk Enclosure hardware installation instructions. The fully populated Power 775 Disk Enclosure consists of 8 STORs of 48 disks, for a total of 384 JBOD disks. Each STOR provides redundant left and right port cards for host server HBA connections (STOR is short for *physical storage group*, meaning the part of the disk enclosure controlled by a pair of port cards). To ensure proper multi-pathing and redundancy, each recovery group server must be connected to each port card using different HBAs. For example, STOR 1 has port cards P1-C4 and P1-C5. Server 1 may be connected to P1-C4 using HBA hba1 and to P1-C5 using HBA hba2; similarly for server 2 and its respective HBAs hba1 and hba2.

GNR provides system administration tools for verifying the correct connectivity of the Power 775 Disk Enclosure, which will be seen later during the operating system preparation.

When the port cards of the Power 775 Disk Enclosure have been cabled to the appropriate HBAs of the two recovery group servers, the Power 775 Disk Enclosure should be powered on and the recovery group servers should be rebooted.

Initial operating system verification

Preparation then continues with the operating system, which must be either AIX 7.1 or Red Hat Enterprise Linux 6.1, and which must be the same on both recovery group servers. It is not necessary to do a complete verification of the Power 775 Disk Enclosure connectivity at this point. Logging in to the servers to perform a quick check that at least some disks have been detected and configured by the operating system will suffice. The operating system device configuration should be examined for the Power 775 Disk Enclosure VPD enclosure type, which is 78AD.001.

One way to quickly verify that AIX has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# lsdev -t ses -F 'name physloc parent' | grep 78AD.001
```

The output should include lines resembling the following:

```
ses12 U78AD.001.000DE37-P1-C4 sas3
```

This is the SAS expander device on port card P1-C4 of the Power 775 Disk Enclosure with serial number 000DE37, together with the SAS protocol device driver `sas3` under which it has been configured. To see what disks have been detected by the SAS protocol driver, use:

```
# lsdev -p sas3
```

The output should include all the disks and port card expanders that successfully configured under the `sas3` SAS protocol driver (which corresponds to the HBA device `mpt2sas3`).

If AIX has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

One way to quickly verify that Red Hat Enterprise Linux has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# grep 78AD.001 /proc/scsi/scsi
```

The output should include lines resembling the following:

```
Vendor: IBM      Model: 78AD-001         Rev: 0150

Further examination of the /proc/scsi/scsi file should reveal contents similar to:
Host: scsi7 Channel: 00 Id: 394 Lun: 00
  Vendor: IBM      Model: ST9600204SS      Rev: 631C
  Type:   Direct-Access                    ANSI  SCSI revision: 06
Host: scsi7 Channel: 00 Id: 395 Lun: 00
  Vendor: IBM      Model: 78AD-001         Rev: 0150
  Type:   Enclosure                        ANSI  SCSI revision: 04
```

The above indicates that a Power 775 Disk Enclosure port card SAS expander and a disk drive have been configured on SCSI host bus 7 (the HBA corresponding to `scsi7`).

As with AIX, if Linux has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

Disabling operating system multi-pathing

When it has been verified that at least some of the Power 775 Disk Enclosure has been configured by the operating system, the next step is to disable any operating system multi-pathing. Because GNR performs its own disk multi-pathing, AIX MPIO (Multiple Path I/O) and Linux DMM (Device Mapper Multipath) must be disabled as appropriate.

To disable AIX MPIO for SAS disks, use:

```
# manage_disk_drivers -d SAS_SCSD -o AIX_non_MPIO
```

To disable Linux DMM, use:

```
# chkconfig --del multipathd
```

Note: This blanket disabling of operating system multi-pathing is appropriate because a Power 775 Disk Enclosure installation provides the only available disk devices to the recovery group servers. When operating system multi-pathing has been disabled, the recovery group servers should be rebooted.

Operating system device attributes

For best performance, the operating system disk device driver should be configured to allow GNR I/O operations to be made with one disk access, rather than being fragmented. Under AIX this is controlled by the `max_transfer` attribute of the HBAs and disk devices. Under Red Hat Enterprise Linux, this is controlled by the disk block device `max_sectors_kb` attribute.

The disk I/O size performed by GNR depends on the strip size of the RAID code of the vdisk NSD. This in turn is related to the vdisk track size and its corresponding GPFS file system block size. The operating system I/O size should be equal to or greater than the largest strip size of the planned vdisk NSDs.

Because GNR stores checksums with each strip, strips have an additional 4 KiB or 8 KiB than might be expected just from the user data (strips containing 2 MiB of user data have an additional 8 KiB; all smaller strips have an additional 4 KiB). The strip size for a replicated vdisk RAID code is equal to the vdisk track size plus the size of the checksum. The strip size for a Reed-Solomon vdisk RAID code is equal to one-eighth of the vdisk track size plus the size of the checksum.

The default `max_transfer` value of 1 MiB under AIX is suitable for GNR vdisk strip sizes under 1 MiB.

The default `max_sectors_kb` value of 512 KiB sectors under Red Hat Enterprise Linux is suitable for GNR vdisk strip sizes under 512 KiB.

However, for vdisk strip sizes greater than 1 MiB under AIX or greater than 512 KiB under Linux, the operating system disk device driver I/O size should be increased for best performance.

The following table indicates the relationship between file system NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and the non-default operating system I/O size for all permitted GNR vdisks. The AIX `max_transfer` attribute is specified in hexadecimal, and the only allowable values greater than the 1 MiB default are 0x200000 (2 MiB) and 0x400000 (4 MiB). Linux `max_sectors_kb` is specified as the desired I/O size in KiB.

Table 46. NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and non-default operating system I/O size for permitted GNR vdisks

NSD block size	vdisk track size	vdisk RAID code	RAID code strip size	AIX max_transfer	Linux max_sectors_kb
256 KiB	256 KiB	3- or 4-way replication	260 KiB	default	260
512 KiB	512 KiB	3- or 4-way replication	516 KiB	default	516

Table 46. NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and non-default operating system I/O size for permitted GNR vdisks (continued)

NSD block size	vdisk track size	vdisk RAID code	RAID code strip size	AIX max_transfer	Linux max_sectors_kb
1 MiB	1 MiB	3- or 4-way replication	1028 KiB	0x200000	1028
2 MiB	2 MiB	3- or 4-way replication	2056 KiB	0x400000	2056
512 KiB	512 KiB	8 + 2p or 8 + 3p	68 KiB	default	default
1 MiB	1 MiB	8 + 2p or 8 + 3p	132 KiB	default	default
2 MiB	2 MiB	8 + 2p or 8 + 3p	260 KiB	default	260
4 MiB	4 MiB	8 + 2p or 8 + 3p	516 KiB	default	516
8 MiB	8 MiB	8 + 2p or 8 + 3p	1028 KiB	0x200000	1028
16 MiB	16 MiB	8 + 2p or 8 + 3p	2056 KiB	0x400000	2056

If the largest strip size of all the vdisk NSDs planned for a GNR installation exceeds the operating system default I/O size, the operating system I/O size should be changed.

AIX

Under AIX, this involves changing the HBA and disk device `max_transfer` size. For an HBA to accommodate an increased `max_transfer` size, the `max_commands` for the HBA will also need to be decreased. With a 0x400000 `max_transfer` size, the four-port HBA requires a `max_commands` value of 124 and the two-port HBA requires a `max_commands` value of 248.

To change the `max_transfer` attribute to 4 MiB for the HBA `mpt2sas0` under AIX, use the following command:

```
# chdev -P -l mpt2sas0 -a max_transfer=0x400000
```

To change the `max_commands` value to 124 for the four-port HBA `mpt2sas0` (AIX device type 001072001410f60), use the following command:

```
# chdev -P -l mpt2sas0 -a max_commands=124
```

To change the `max_commands` value to 248 for the two-port HBA `mpt2sas0` (AIX device type 001072001410ea0), use the following command:

```
# chdev -P -l mpt2sas0 -a max_commands=248
```

Repeat the previous commands for each HBA.

Changing the hdisk `max_transfer` attribute requires changing its default value for AIX device type nonmpioscsd disks. It is not sufficient to change the `max_transfer` for individual hdisks, because performing disk replacement deletes and recreates hdisk objects.

To change the default hdisk `max_transfer` attribute for type nonmpioscsd hdisks, use the following command:

```
# chdef -a max_transfer=0x400000 -c disk -s sas -t nonmpioscsd
```

The new `max_transfer` and `max_commands` values will not take effect until AIX reconfigures the HBAs and hdisks. This can be done either by rebooting the recovery group server, or by deconfiguring (but not removing) and then reconfiguring the affected HBAs. The `max_transfer` and `max_commands` attribute are recorded in the CuAt ODM class and will persist across reboots.

Linux

Under Linux, the `max_sectors_kb` I/O size is reset to the default each time a disk block device is configured. This means that upon reboot and upon adding or replacing disks, a desired nondefault I/O size must be explicitly set. Because `max_sectors_kb` is dynamically reconfigurable, GPFS Native RAID manages this setting under Linux.

The value of `max_sectors_kb` under Linux is set on all the disks in a recovery group when GPFS Native RAID begins serving the recovery group, and on disks that are configured as part of GPFS Native RAID disk replacement.

The default `max_sectors_kb` set by GPFS Native RAID is 4096, which is large enough for any of the strip sizes listed in Table 46 on page 421. It can be changed to exactly match the largest strip size in use by GPFS Native RAID by setting the GPFS `nsdRAIDBlockDeviceMaxSectorsKB` configuration parameter.

To set the `max_sectors_kb` value used by GPFS Native RAID to 2056, use the following command:

```
# mmchconfig nsdRAIDBlockDeviceMaxSectorsKB=2056
```

The new value will take effect the next time GPFS is started. To have the new value take effect immediately, append the `-i` option to the above command.

For optimal performance, additional device attributes may need to be changed (for example, the HBA and block device command queue depths); consult the operating system documentation for the device attributes.

Verifying that a Power 775 Disk Enclosure is configured correctly

When a superficial inspection indicates that the Power 775 Disk Enclosure has been configured on the recovery group servers, and especially after operating system multi-pathing has been disabled, it is necessary to perform a thorough discovery of the disk topology on each server.

To proceed, GPFS must be installed on the recovery group servers, and they should be members of the same GPFS cluster. Consult the *IBM Storage Scale: Administration Guide* for instructions for creating a GPFS cluster.

GNR provides tools in `/usr/lpp/mmfs/samples/vdisk` for collecting and collating information on any attached Power 775 Disk Enclosure and for verifying that the detected topology is correct. The `mmgetpdisktopology` command examines the operating system's list of connected devices and produces a colon-delimited database with a line for each discovered Power 775 Disk Enclosure physical disk, port card expander device, and HBA. `mmgetpdisktopology` should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called `topsummary` concisely summarizes the output of the `mmgetpdisktopology` command.

Create a directory in which to work, and then capture the output of the `mmgetpdisktopology` command from each of the two intended recovery group server nodes:

```
# mkdir p7ihde
# cd p7ihde
# ssh server1 /usr/lpp/mmfs/bin/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/bin/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes (`server1` example shown):

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
P7IH-DE enclosures found: DE00022
Enclosure DE00022:
Enclosure DE00022 STOR P1-C4/P1-C5 sees both portcards: P1-C4 P1-C5
Portcard P1-C4: ses0[0150]/mpt2sas0/24 diskset "37993" ses1[0150]/mpt2sas0/24 diskset "18793"
Portcard P1-C5: ses4[0150]/mpt2sas1/24 diskset "37993" ses5[0150]/mpt2sas1/24 diskset "18793"
Enclosure DE00022 STOR P1-C4/P1-C5 sees 48 disks
Enclosure DE00022 STOR P1-C12/P1-C13 sees both portcards: P1-C12 P1-C13
Portcard P1-C12: ses8[0150]/mpt2sas2/24 diskset "40657" ses9[0150]/mpt2sas2/24 diskset "44382"
Portcard P1-C13: ses12[0150]/mpt2sas3/24 diskset "40657" ses13[0150]/mpt2sas3/24 diskset "44382"
```

```
Enclosure DE00022 STOR P1-C12/P1-C13 sees 48 disks
Enclosure DE00022 STOR P1-C20/P1-C21 sees both portcards: P1-C20 P1-C21
Portcard P1-C20: ses16[0150]/mpt2sas4/24 diskset "04091" ses17[0150]/mpt2sas4/24 diskset "31579"
Portcard P1-C21: ses20[0150]/mpt2sas5/24 diskset "04091" ses21[0150]/mpt2sas5/24 diskset "31579"
Enclosure DE00022 STOR P1-C20/P1-C21 sees 48 disks
Enclosure DE00022 STOR P1-C28/P1-C29 sees both portcards: P1-C28 P1-C29
Portcard P1-C28: ses24[0150]/mpt2sas6/24 diskset "64504" ses25[0150]/mpt2sas6/24 diskset "62361"
Portcard P1-C29: ses28[0150]/mpt2sas7/24 diskset "64504" ses29[0150]/mpt2sas7/24 diskset "62361"
Enclosure DE00022 STOR P1-C28/P1-C29 sees 48 disks
Enclosure DE00022 STOR P1-C60/P1-C61 sees both portcards: P1-C60 P1-C61
Portcard P1-C60: ses30[0150]/mpt2sas7/24 diskset "10913" ses31[0150]/mpt2sas7/24 diskset "52799"
Portcard P1-C61: ses26[0150]/mpt2sas6/24 diskset "10913" ses27[0150]/mpt2sas6/24 diskset "52799"
Enclosure DE00022 STOR P1-C60/P1-C61 sees 48 disks
Enclosure DE00022 STOR P1-C68/P1-C69 sees both portcards: P1-C68 P1-C69
Portcard P1-C68: ses22[0150]/mpt2sas5/24 diskset "50112" ses23[0150]/mpt2sas5/24 diskset "63400"
Portcard P1-C69: ses18[0150]/mpt2sas4/24 diskset "50112" ses19[0150]/mpt2sas4/24 diskset "63400"
Enclosure DE00022 STOR P1-C68/P1-C69 sees 48 disks
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
Enclosure DE00022 STOR P1-C84/P1-C85 sees both portcards: P1-C84 P1-C85
Portcard P1-C84: ses6[0150]/mpt2sas1/24 diskset "13325" ses7[0150]/mpt2sas1/24 diskset "10443"
Portcard P1-C85: ses2[0150]/mpt2sas0/24 diskset "13325" ses3[0150]/mpt2sas0/24 diskset "10443"
Enclosure DE00022 STOR P1-C84/P1-C85 sees 48 disks
Carrier location P1-C79-D4 appears only on the portcard P1-C77 path
Enclosure DE00022 sees 384 disks

mpt2sas7[1005470001] U78A9.001.9998884-P1-C1 DE00022 STOR 4 P1-C29 (ses28 ses29) STOR 5 P1-C60
(ses30 ses31)
mpt2sas6[1005470001] U78A9.001.9998884-P1-C3 DE00022 STOR 4 P1-C28 (ses24 ses25) STOR 5 P1-C61
(ses26 ses27)
mpt2sas5[1005470001] U78A9.001.9998884-P1-C5 DE00022 STOR 3 P1-C21 (ses20 ses22) STOR 6 P1-C68
(ses21 ses23)
mpt2sas4[1005470001] U78A9.001.9998884-P1-C7 DE00022 STOR 3 P1-C20 (ses16 ses17) STOR 6 P1-C69
(ses18 ses19)
mpt2sas3[1005470001] U78A9.001.9998884-P1-C9 DE00022 STOR 2 P1-C13 (ses12 ses13) STOR 7 P1-C76
(ses14 ses15)
mpt2sas2[1005470001] U78A9.001.9998884-P1-C11 DE00022 STOR 2 P1-C12 (ses8 ses9) STOR 7 P1-C77
(ses10 ses11)
mpt2sas1[1005470001] U78A9.001.9998884-P1-C13 DE00022 STOR 1 P1-C5 (ses4 ses5) STOR 8 P1-C84
(ses6 ses7)
mpt2sas0[1005470001] U78A9.001.9998884-P1-C15 DE00022 STOR 1 P1-C4 (ses0 ses1) STOR 8 P1-C85
(ses2 ses3)
```

In the preceding output, the Power 775 Disk Enclosure with serial number DE00022 is discovered, together with its eight individual STORs and the component port cards, port card expanders (with their firmware levels in brackets), and physical disks. One minor discrepancy is noted: The physical disk in location P1-C79-D4 is only seen over one of the two expected HBA paths. This can also be seen in the output for the STOR with port cards P1-C76 and P1-C77:

```
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
```

Here the connection through port card P1-C76 sees just 23 disks on the expander `ses14` and all 24 disks on the expander `ses15`, while the connection through port card P1-C77 sees all 24 disks on each of the expanders `ses10` and `ses11`. The "disksets" that are reached over the expanders are identified by a checksum of the unique SCSI WWNs of the physical disks that are present; equal disksets represent the same collection of physical disks.

The preceding discrepancy can either be corrected or ignored, as it is probably due to a poorly seated or defective port on the physical disk. The disk is still present on the other port.

If other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), they should be corrected before proceeding.

The HBAs (firmware levels in brackets) are also listed with their slot location codes to show the cabling pattern. Each HBA sees two STORs, and each STOR is seen by two different HBAs, which provides the multiple paths and redundancy required by a correct Power 775 Disk Enclosure installation.

This output can be compared to the hardware cabling specification to verify that the disk enclosure is connected correctly.

The `server2.top` topology database should also be examined with the `topsummary` sample script and verified to be correct.

When the Power 775 Disk Enclosure topologies are verified to be correct on both intended recovery group server nodes, the recommended recovery group configuration can be created using GNR commands.

Creating recovery groups on a Power 775 Disk Enclosure

Configuring GPFS nodes to be recovery group servers

Before a GPFS node can create and serve recovery groups, it must be configured with a vdisk track cache. This is accomplished by setting the **nsdRAIDTracks** configuration parameter.

nsdRAIDTracks is the GPFS configuration parameter essential to define a GPFS cluster node as a recovery group server. It specifies the number of vdisk tracks of which the attributes will be held in memory by the GPFS daemon on the recovery group server.

The actual contents of the vdisk tracks, the user data and the checksums, are stored in the standard GPFS page pool. Therefore, the size of the GPFS page pool configured on a recovery group server should be considerable, on the order of tens of gigabytes. The amount of page pool dedicated to hold vdisk track data is governed by the **nsdRAIDBufferPoolSizePct** parameter, which defaults to 50%. In practice, a recovery group server will not need to use the GPFS page pool for any significant amount of standard file caching, and the **nsdRAIDBufferPoolSizePct** value can be increased to 80%. Also applicable, since a recovery group server is by definition an NSD server, is the **nsdBufSpace** parameter, which defaults to 30% of page pool. Since the vdisk buffer pool doubles as the NSD buffer spool, the **nsdBufSpace** parameter should be decreased to its minimum of 10%. Together these values leave only 10% of the page pool for application program file cache, but this should not be a problem as a recovery group server should not be running application programs.

In this example, the recovery group servers will be configured to cache the information on 16384 vdisk tracks and to have 64 GiB of page pool, of which 80% will be used for vdisk data. Once the configuration changes are made, the servers will need to be restarted.

```
# mmchconfig nsdRAIDTracks=16384,nsdRAIDBufferPoolSizePct=80,nsdBufSpace=10,pagepool=64G -N
server1,server2
# mmshutdown -N server1,server2
# mmstartup -N server1,server2
```

Defining the recovery group layout

The definition of recovery groups on a Power 775 Disk Enclosure is dictated by the architecture and cabling of the disk enclosure. Two servers sharing a Power 775 Disk Enclosure implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each STOR should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks and carriers in the top halves of the eight STORs, and one on the bottom halves. Since the disks in a STOR are placed four to a removable carrier, thereby having a common point of failure, each disk in a carrier should belong to one of four different declustered arrays. Should a carrier fail or be removed, then each declustered array will only suffer the loss of one disk. There are four SSDs distributed among the top set of carriers, and four in the bottom set of carriers. These groups of four SSDs will make up the vdisk log declustered arrays in their respective halves.

GNR provides a tool that understands the layout of the Power 775 Disk Enclosure and will automatically generate the **mmcrrecoverygroup** stanza files for creating the top and bottom recovery groups. `/usr/lpp/mmfs/samples/vdisk/mkp7rginput`, when supplied with output of the **mmgetpdisktopology** command, will create recovery group stanza files for the top and bottom halves of each Power 775 Disk Enclosure found in the topology.

Each recovery group server, though it may see the same functional disk enclosure topology, will almost certainly differ in the particulars of which disk device names (e.g., /dev/rhdisk77 on AIX or /dev/sdax on Linux) refer to which physical disks in what disk enclosure location.

There are two possibilities then for creating the recovery group stanza files and the recovery groups themselves:

Alternative 1:
Generate the recovery group stanza files and create the recovery groups from the perspective of just one of the servers as if that server were to be primary for both recovery groups, and then use the **mmchrecoverygroup** command to swap the primary and backup servers for one of the recovery groups

Alternative 2:
Generate the recovery group stanza files for each server's primary recovery group using the primary server's topology file.

This example will show both alternatives.

Creating the recovery groups, alternative 1
To create the recovery group input stanza files from the perspective of server1, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

This will create two files for each disk enclosure present in the server1 topology; in this case, DE00022TOP.server1 for the top half of disk enclosure DE00022 and DE00022BOT.server2 for the bottom half. (An extra file, DEXXXXXbad, may be created if any discrepancies are present in the topology; if such a file is created by **mkp7rginput**, it should be examined and the discrepancies corrected.)

The recovery group stanza files will follow the recommended best practice for the Power 775 Disk Enclosure of defining in each half of the disk enclosure a separate declustered array of 4 SSDs for recovery group transaction logging, and four file system data declustered arrays using the regular HDDs according to which of the four disk enclosure carrier slots each HDD resides in.

The defaults are accepted for other recovery group declustered array parameters such as scrub duration, spare space, and disk replacement policy.

The stanza file will look something like this:

```
# head DE00022TOP.server1
%pdisk: pdiskName=c081d1
        device=/dev/hdisk10
        da=DA1
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c065d1
        device=/dev/hdisk211
        da=DA1
        nPathActive=2
        nPathTotal=4
%pdisk: pdiskName=c066d1
        device=/dev/hdisk259
        da=DA1
        nPathActive=2
        nPathTotal=4
```

All the pdisk stanzas for declustered array DA1 will be listed first, followed by those for DA2, DA3, DA4, and the LOG declustered array. The pdisk names will indicate the carrier and disk location in which the physical disk resides. Notice that only one block device path to the disk is given; the second path will be discovered automatically soon after the recovery group is created.

Now that the DE00022TOP.server1 and DE00022BOT.server1 stanza files have been created from the perspective of recovery group server node server1, these two recovery groups can be created using two separate invocations of the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note that both recovery groups were created with `server1` as primary and `server2` as backup. It is now necessary to swap the primary and backup servers for DE00022BOT using the **mmchrecoverygroup** command:

```
# mmchrecoverygroup DE00022BOT --servers server2,server1
mmchrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

GNR will automatically discover the appropriate disk devices on `server2`.

Creating the recovery groups, alternative 2

To create the recovery groups from the start with the intended primary and backup servers, the stanza files from both server topologies will need to be created.

To create the `server1` recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

To create the `server2` recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server2.top
```

These two commands will result in four stanza files: DE00022TOP.`server1`, DE00022BOT.`server1`, DE00022TOP.`server2`, and DE00022BOT.`server2`. (As in alternative 1, if any files named DEXXXXXbad are created, they should be examined and the errors within should be corrected.)

The DE00022TOP recovery group must then be created using `server1` as the primary and the DE00022TOP.`server1` stanza file. The DE00022BOT recovery group must be created using `server2` as the primary and the DE00022BOT.`server2` stanza file.

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server2 --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Because each recovery group was created using the intended primary server and the stanza file for that server, it is not necessary to swap the primary and backup servers.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created:

```
# mmlsrecoverygroup DE00022TOP -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022TOP                   5       0     192

 declustered   needs                            replace                scrub       background
activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task
progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------
```

```
-------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive
0%  low

                                        declustered
 vdisk               RAID code             array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------

 active recovery group server                      servers
 ------------------------------------------------  -------
 server1                                           server1,server2

# mmlsrecoverygroup DE00022BOT -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022BOT                   5       0     192

 declustered   needs                           replace                scrub       background
activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task
progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------
-------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive
0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive
0%  low

                                        declustered
 vdisk               RAID code             array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------

 active recovery group server                      servers
 ------------------------------------------------  -------
 server1                                           server2,server1
```

Notice that the vdisk sections of the newly created recovery groups are empty; the next step is to create the vdisks.

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

Each recovery group requires a single log vdisk for recording RAID updates and diagnostic information. This is internal to the recovery group, cannot be used for user data, and should be the only vdisk in the LOG declustered array. The log vdisks in this example use 3-way replication in order to fit in the LOG declustered array, which contains 4 SSDs and spare space equivalent to one disk.

Data vdisks are required to be defined in the four data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics: one using 4-way replication and a 1 MiB block size and a total vdisk size of 250 GiB suitable for file system metadata, and one using Reed-Solomon 8 + 3p encoding and a 16 MiB block size suitable for file system data. The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template.

In this example, a single stanza file, `mmcrvdisk.DE00022ALL`, is used. The single file contains the specifications for all the vdisks in both the DE00022TOP and DE00022BOT recovery groups. Here is what the example stanza file for use with **mmcrvdisk** should look like:

```
# cat mmcrvdisk.DE00022ALL
%vdisk: vdiskName=DE00022TOPLOG
        rg=DE00022TOP
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022BOTLOG
        rg=DE00022BOT
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022TOPDA1META
        rg=DE00022TOP
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022TOPDA1DATA
        rg=DE00022TOP
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
%vdisk: vdiskName=DE00022BOTDA1META
        rg=DE00022BOT
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022BOTDA1DATA
        rg=DE00022BOT
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
[DA2, DA3, DA4 vdisks omitted.]
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as **metadataOnly** and for placement in the `system` storage pool, and the file system data vdisks are flagged for eventual **dataOnly** usage in the `data` storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see “Creating the GPFS file system” on page 431.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual `failureGroup=22` value for the NSDs on the file system vdisks is the same for vdisks in both the DE00022TOP and DE00022BOT recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the disk enclosure DE00022, and GPFS should be informed of this through a distinct failure group designation for each disk

enclosure. It is up to the GPFS system administrator to decide upon the failure group numbers for each Power 775 Disk Enclosure in the GPFS cluster.

To create the vdisks specified in the `mmcrvdisk.DE00022ALL` file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisk.DE00022ALL
mmcrvdisk: [I] Processing vdisk DE00022TOPLOG
mmcrvdisk: [I] Processing vdisk DE00022BOTLOG
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4DATA
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the vdisks can be verified through the **mmlsvdisk** command (the **mmlsrecoverygroup** command can also be used):

```
# mmlsvdisk
                                                    declustered  block size
 vdisk name          RAID code        recovery group     array       in KiB    remarks
 ------------------  ---------------  ------------------ -----------  ----------  -------
 DE00022BOTDA1DATA   8+3p             DE00022BOT         DA1               16384
 DE00022BOTDA1META   4WayReplication  DE00022BOT         DA1                1024
 DE00022BOTDA2DATA   8+3p             DE00022BOT         DA2               16384
 DE00022BOTDA2META   4WayReplication  DE00022BOT         DA2                1024
 DE00022BOTDA3DATA   8+3p             DE00022BOT         DA3               16384
 DE00022BOTDA3META   4WayReplication  DE00022BOT         DA3                1024
 DE00022BOTDA4DATA   8+3p             DE00022BOT         DA4               16384
 DE00022BOTDA4META   4WayReplication  DE00022BOT         DA4                1024
 DE00022BOTLOG       3WayReplication  DE00022BOT         LOG                1024  log
 DE00022TOPDA1DATA   8+3p             DE00022TOP         DA1               16384
 DE00022TOPDA1META   4WayReplication  DE00022TOP         DA1                1024
 DE00022TOPDA2DATA   8+3p             DE00022TOP         DA2               16384
 DE00022TOPDA2META   4WayReplication  DE00022TOP         DA2                1024
 DE00022TOPDA3DATA   8+3p             DE00022TOP         DA3               16384
 DE00022TOPDA3META   4WayReplication  DE00022TOP         DA3                1024
 DE00022TOPDA4DATA   8+3p             DE00022TOP         DA4               16384
 DE00022TOPDA4META   4WayReplication  DE00022TOP         DA4                1024
 DE00022TOPLOG       3WayReplication  DE00022TOP         LOG                1024  log
```

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which GPFS builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisk.DE00022ALL
mmcrnsd: Processing disk DE00022TOPDA1META
mmcrnsd: Processing disk DE00022TOPDA1DATA
mmcrnsd: Processing disk DE00022TOPDA2META
mmcrnsd: Processing disk DE00022TOPDA2DATA
mmcrnsd: Processing disk DE00022TOPDA3META
mmcrnsd: Processing disk DE00022TOPDA3DATA
mmcrnsd: Processing disk DE00022TOPDA4META
mmcrnsd: Processing disk DE00022TOPDA4DATA
mmcrnsd: Processing disk DE00022BOTDA1META
mmcrnsd: Processing disk DE00022BOTDA1DATA
mmcrnsd: Processing disk DE00022BOTDA2META
```

```
mmcrnsd: Processing disk DE00022BOTDA2DATA
mmcrnsd: Processing disk DE00022BOTDA3META
mmcrnsd: Processing disk DE00022BOTDA3DATA
mmcrnsd: Processing disk DE00022BOTDA4META
mmcrnsd: Processing disk DE00022BOTDA4DATA
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the recovery group log vdisks are omitted from NSD processing.

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

```
# mmcrfs gpfs -F mmcrvdisk.DE00022ALL -B 16m --metadata-block-size 1m -T /gpfs -A no

The following disks of gpfs will be formatted on node c250f09c01ap05.ppd.pok.ibm.com:
    DE00022TOPDA1META: size 262163456 KB
    DE00022TOPDA1DATA: size 8395522048 KB
    DE00022TOPDA2META: size 262163456 KB
    DE00022TOPDA2DATA: size 8395522048 KB
    DE00022TOPDA3META: size 262163456 KB
    DE00022TOPDA3DATA: size 8395522048 KB
    DE00022TOPDA4META: size 262163456 KB
    DE00022TOPDA4DATA: size 8395522048 KB
    DE00022BOTDA1META: size 262163456 KB
    DE00022BOTDA1DATA: size 8395522048 KB
    DE00022BOTDA2META: size 262163456 KB
    DE00022BOTDA2DATA: size 8395522048 KB
    DE00022BOTDA3META: size 262163456 KB
    DE00022BOTDA3DATA: size 8395522048 KB
    DE00022BOTDA4META: size 262163456 KB
    DE00022BOTDA4DATA: size 8395522048 KB
Formatting file system ...
Disks up to size 2.5 TB can be added to storage pool 'system'.
Disks up to size 79 TB can be added to storage pool 'data'.
Creating Inode File
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
Formatting Allocation Map for storage pool 'data'
Completed creation of file system /dev/gpfs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system by using the **mmchpolicy** command:

```
# mmchpolicy gpfs policy -I yes
Validated policy `policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude
Rules,
        0 List Rules, 0 External Pool/List Rules
Policy `policy' installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, which is built on a Power 775 Disk Enclosure by using two recovery groups, two recovery group servers, eight file system metadata vdisk NSDs and eight file system data vdisk NSDs, can now be mounted and placed into service:

```
# mmmount gpfs -a
```

Replacing failed disks in a Power 775 Disk Enclosure recovery group: a sample scenario

The scenario presented here shows how to detect and replace failed disks in a recovery group built on a Power 775 Disk Enclosure.

Detecting failed disks in your enclosure

Assume a fully-populated Power 775 Disk Enclosure (serial number 000DE37) on which the following two recovery groups are defined:

- `000DE37TOP` containing the disks in the top set of carriers
- `000DE37BOT` containing the disks in the bottom set of carriers

Each recovery group contains the following:

- one log declustered array (LOG)
- four data declustered arrays (DA1, DA2, DA3, DA4)

The data declustered arrays are defined according to Power 775 Disk Enclosure best practice as follows:

- 47 pdisks per data declustered array
- each member pdisk from the same carrier slot
- default disk replacement threshold value set to 2

The replacement threshold of 2 means that GNR will only require disk replacement when two or more disks have failed in the declustered array; otherwise, rebuilding onto spare space or reconstruction from redundancy will be used to supply affected data.

This configuration can be seen in the output of `mmlsrecoverygroup` for the recovery groups, shown here for `000DE37TOP`:

```
# mmlsrecoverygroup 000DE37TOP -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                            replace                scrub       background
activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task
progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------
-------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub
63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub
19%  low
 DA3          yes           2      47       2          2       0   B   14 days  rebuild-2r
48%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub
33%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub
87%  low

                                          declustered
 vdisk               RAID code              array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------
 000DE37TOPLOG       3WayReplication     LOG            4144 MiB  log
 000DE37TOPDA1META   4WayReplication     DA1             250 GiB
 000DE37TOPDA1DATA   8+3p                DA1              17 TiB
```

```
  000DE37TOPDA2META   4WayReplication     DA2             250 GiB
  000DE37TOPDA2DATA   8+3p                DA2              17 TiB
  000DE37TOPDA3META   4WayReplication     DA3             250 GiB
  000DE37TOPDA3DATA   8+3p                DA3              17 TiB
  000DE37TOPDA4META   4WayReplication     DA4             250 GiB
  000DE37TOPDA4DATA   8+3p                DA4              17 TiB

  active recovery group server                    servers
  ----------------------------------------------  -------
  server1                                         server1,server2
```

The indication that disk replacement is called for in this recovery group is the value of `yes` in the `needs service` column for declustered array DA3.

The fact that DA3 (the declustered array on the disks in carrier slot 3) is undergoing rebuild of its RAID tracks that can tolerate two strip failures is by itself not an indication that disk replacement is required; it merely indicates that data from a failed disk is being rebuilt onto spare space. Only if the replacement threshold has been met will disks be marked for replacement and the declustered array marked as needing service.

GNR provides several indications that disk replacement is required:

- entries in the AIX error report or the Linux syslog
- the pdReplacePdisk callback, which can be configured to run an administrator-supplied script at the moment a pdisk is marked for replacement
- the POWER7® cluster event notification TEAL agent, which can be configured to send disk replacement notices when they occur to the POWER7 cluster EMS
- the output from the following commands, which may be performed from the command line on any GPFS cluster node (see the examples that follow):
  1. `mmlsrecoverygroup` with the `-L` flag shows `yes` in the `needs service` column
  2. `mmlsrecoverygroup` with the `-L` and `--pdisk` flags; this shows the states of all pdisks, which may be examined for the `replace` pdisk state
  3. `mmlspdisk` with the `--replace` flag, which lists only those pdisks that are marked for replacement

Note: Because the output of `mmlsrecoverygroup -L --pdisk` for a fully-populated disk enclosure is very long, this example shows only some of the pdisks (but includes those marked for replacement).

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                  5       9     192

 declustered   needs                            replace                 scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2   3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2   3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      47       2          2      0   B   14 days  rebuild-2r  68%  low
 DA4          no            2      47       2          2   3072 MiB   14 days  scrub       34%  low
 LOG          no            1       4       1          1    546 GiB   14 days  scrub       87%  low

                    n. active,   declustered                user     state,
pdisk               total paths     array     free space   condition  remarks
-----------------   -----------  -----------  ----------  ----------- -------
[...]
c014d1                2,  4      DA1             62 GiB  normal      ok
c014d2                2,  4      DA2            279 GiB  normal      ok
c014d3                0,  0      DA3            279 GiB  replaceable dead/systemDrain/noRGD/noVCD/
replace
c014d4                2,  4      DA4             12 GiB  normal      ok
[...]
c018d1                2,  4      DA1             24 GiB  normal      ok
c018d2                2,  4      DA2             24 GiB  normal      ok
c018d3                2,  4      DA3            558 GiB  replaceable dead/systemDrain/noRGD/noVCD/
noData/replace
```

```
c018d4               2,  4      DA4             12 GiB  normal      ok
[...]
```

The preceding output shows that the following pdisks are marked for replacement:

- c014d3 in DA3
- c018d3 in DA3

The naming convention used during recovery group creation indicates that these are the disks in slot 3 of carriers 14 and 18. To confirm the physical locations of the failed disks, use the `mmlspdisk` command to list information about those pdisks in declustered array DA3 of recovery group `000DE37TOP` that are marked for replacement:

```
# mmlspdisk 000DE37TOP --declustered-array DA3 --replace
pdisk:
   replacementPriority = 1.00
   name = "c014d3"
   device = "/dev/rhdisk158,/dev/rhdisk62"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/replace"
   .
   .
   .

pdisk:
   replacementPriority = 1.00
   name = "c018d3"
   device = "/dev/rhdisk630,/dev/rhdisk726"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/noData/replace"
   .
   .
   .
```

The preceding location code attributes confirm the pdisk naming convention:

Disk	Location code	Interpretation
pdisk c014d3	78AD.001.000DE37-C14-D3	Disk 3 in carrier 14 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37
pdisk c018d3	78AD.001.000DE37-C18-D3	Disk 3 in carrier 18 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37

Replacing the failed disks in a Power 775 Disk Enclosure recovery group

Note: In this example, it is assumed that two new disks with the appropriate Field Replaceable Unit (FRU) code, as indicated by the `fru` attribute (74Y4936 in this case), have been obtained as replacements for the failed pdisks c014d3 and c018d3.

Replacing each disk is a three-step process:

1. Using the `mmchcarrier` command with the `--release` flag to suspend use of the other disks in the carrier and to release the carrier.
2. Removing the carrier and replacing the failed disk within with a new one.
3. Using the `mmchcarrier` command with the `--replace` flag to resume use of the suspended disks and to begin use of the new disk.

GNR assigns a priority to pdisk replacement. Disks with smaller values for the `replacementPriority` attribute should be replaced first. In this example, the only failed disks are in DA3 and both have the same `replacementPriority`.

Disk c014d3 is chosen to be replaced first.

1. To release carrier 14 in disk enclosure 000DE37:

```
# mmchcarrier 000DE37TOP --release --pdisk c014d3
  [I] Suspending pdisk c014d1 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D1.
  [I] Suspending pdisk c014d2 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D2.
  [I] Suspending pdisk c014d3 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D3.
  [I] Suspending pdisk c014d4 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D4.
  [I] Carrier released.

    - Remove carrier.
    - Replace disk in location 78AD.001.000DE37-C14-D3 with FRU 74Y4936.
    - Reinsert carrier.
    - Issue the following command:

        mmchcarrier 000DE37TOP --replace --pdisk 'c014d3'

      Repair timer is running.  Perform the above within 5 minutes
      to avoid pdisks being reported as missing.
```

   GNR issues instructions as to the physical actions that must be taken. Note that disks may be suspended only so long before they are declared missing; therefore the mechanical process of physically performing disk replacement must be accomplished promptly.

   Use of the other three disks in carrier 14 has been suspended, and carrier 14 is unlocked. The identify lights for carrier 14 and for disk 3 are on.

2. Carrier 14 should be unlatched and removed. The failed disk 3, as indicated by the internal identify light, should be removed, and the new disk with FRU 74Y4936 should be inserted in its place. Carrier 14 should then be reinserted and the latch closed.

3. To finish the replacement of pdisk c014d3:

```
# mmchcarrier 000DE37TOP --replace --pdisk c014d3
[I] The following pdisks will be formatted on node server1:
    /dev/rhdisk354
[I] Pdisk c014d3 of RG 000DE37TOP successfully replaced.
[I] Resuming pdisk c014d1 of RG 000DE37TOP.
[I] Resuming pdisk c014d2 of RG 000DE37TOP.
[I] Resuming pdisk c014d3#162 of RG 000DE37TOP.
[I] Resuming pdisk c014d4 of RG 000DE37TOP.
[I] Carrier resumed.
```

When the `mmchcarrier --replace` command returns successfully, GNR has resumed use of the other 3 disks. The failed pdisk may remain in a temporary form (indicated here by the name c014d3#162) until all data from it has been rebuilt, at which point it is finally deleted. The new replacement disk, which has assumed the name c014d3, will have RAID tracks rebuilt and rebalanced onto it. Notice that only one block device name is mentioned as being formatted as a pdisk; the second path will be discovered in the background.

This can be confirmed with `mmlsrecoverygroup -L --pdisk`:

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     193

 declustered   needs                            replace                 scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      48       2          2       0   B   14 days  rebuild-2r  89%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       34%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                   n. active,   declustered                 user     state,
pdisk              total paths     array     free space   condition  remarks
```

```
-----------------   -----------  -----------  ----------  ----------- -------
[...]
c014d1                2,  4      DA1             23 GiB  normal      ok
c014d2                2,  4      DA2             23 GiB  normal      ok
c014d3                2,  4      DA3            550 GiB  normal      ok
c014d3#162            0,  0      DA3            543 GiB  replaceable dead/adminDrain/noRGD/noVCD/noPath
c014d4                2,  4      DA4             23 GiB  normal      ok
[...]
c018d1                2,  4      DA1             24 GiB  normal      ok
c018d2                2,  4      DA2             24 GiB  normal      ok
c018d3                0,  0      DA3            558 GiB  replaceable dead/systemDrain/noRGD/noVCD/
noData/replace
c018d4                2,  4      DA4             23 GiB  normal      ok
[...]
```

Notice that the temporary pdisk `c014d3#162` is counted in the total number of pdisks in declustered array DA3 and in the recovery group, until it is finally drained and deleted.

Notice also that pdisk `c018d3` is still marked for replacement, and that DA3 still needs service. This is because GNR replacement policy expects all failed disks in the declustered array to be replaced once the replacement threshold is reached. The `replace` state on a pdisk is not removed when the total number of failed disks goes under the threshold.

Pdisk `c018d3` is replaced following the same process.

1. Release carrier 18 in disk enclosure `000DE37`:

```
# mmchcarrier 000DE37TOP --release --pdisk c018d3
   [I] Suspending pdisk c018d1 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D1.
   [I] Suspending pdisk c018d2 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D2.
   [I] Suspending pdisk c018d3 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D3.
   [I] Suspending pdisk c018d4 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D4.
   [I] Carrier released.

     - Remove carrier.
     - Replace disk in location 78AD.001.000DE37-C18-D3 with FRU 74Y4936.
     - Reinsert carrier.
     - Issue the following command:

         mmchcarrier 000DE37TOP --replace --pdisk 'c018d3'

       Repair timer is running.  Perform the above within 5 minutes
       to avoid pdisks being reported as missing.
```

2. Unlatch and remove carrier 18, remove and replace failed disk 3, reinsert carrier 18, and close the latch.
3. To finish the replacement of pdisk `c018d3`:

```
# mmchcarrier 000DE37TOP --replace --pdisk c018d3

   [I] The following pdisks will be formatted on node server1:
       /dev/rhdisk674
   [I] Pdisk c018d3 of RG 000DE37TOP successfully replaced.
   [I] Resuming pdisk c018d1 of RG 000DE37TOP.
   [I] Resuming pdisk c018d2 of RG 000DE37TOP.
   [I] Resuming pdisk c018d3#166 of RG 000DE37TOP.
   [I] Resuming pdisk c018d4 of RG 000DE37TOP.
   [I] Carrier resumed.
```

Running `mmlsrecoverygroup` again will confirm the second replacement:

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                 5       9     192

 declustered   needs                           replace                 scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       64%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       22%  low
```

```
 DA3          no            2      47       2          2    2048 MiB   14 days  rebalance   12%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       36%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       89%  low

                     n. active,   declustered                 user     state,
pdisk                total paths     array     free space   condition  remarks
-----------------    -----------  -----------  ----------  ----------- -------
 [...]
 c014d1                2,  4       DA1             23 GiB  normal      ok
 c014d2                2,  4       DA2             23 GiB  normal      ok
 c014d3                2,  4       DA3            271 GiB  normal      ok
 c014d4                2,  4       DA4             23 GiB  normal      ok
 [...]
 c018d1                2,  4       DA1             24 GiB  normal      ok
 c018d2                2,  4       DA2             24 GiB  normal      ok
 c018d3                2,  4       DA3            542 GiB  normal      ok
 c018d4                2,  4       DA4             23 GiB  normal      ok
 [...]
```

Notice that both temporary pdisks have been deleted. This is because c014d3#162 has finished draining, and because pdisk c018d3#166 had, before it was replaced, already been completely drained (as evidenced by the noData flag). Declustered array DA3 no longer needs service and once again contains 47 pdisks, and the recovery group once again contains 192 pdisks.

Accessibility features for IBM Storage Scale RAID

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Storage Scale RAID:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

IBM Documentation, and its related publications, are accessibility-enabled.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center (www.ibm.com/able) for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21,

Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. 30ZA/Building 707
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment or a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Intel is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat and Ansible® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

IBM Privacy Policy

At IBM we recognize the importance of protecting your personal information and are committed to processing it responsibly and in compliance with applicable data protection laws in all countries in which IBM operates.

Visit the IBM Privacy Policy for additional information on this topic at https://www.ibm.com/privacy/details/us/en/.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You can reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You cannot distribute, display, or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You can reproduce, distribute, and display these publications solely within your enterprise provided that all proprietary notices are preserved. You cannot make derivative works of these publications, or reproduce, distribute, or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses, or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions that are granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or as determined by IBM, the above instructions are not being properly followed.

You cannot download, export, or reexport this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Glossary

This glossary provides terms and definitions for the ESS solution.

The following cross-references are used in this glossary:

- *See* refers you from a non-preferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

For other terms and definitions, see the IBM Terminology website (opens in new window):

http://www.ibm.com/software/globalization/terminology

B

building block
A pair of servers with shared disk enclosures attached.

BOOTP
See *Bootstrap Protocol (BOOTP)*.

Bootstrap Protocol (BOOTP)
A computer networking protocol that is used in IP networks to automatically assign an IP address to network devices from a configuration server.

C

CEC
See *central processor complex (CPC)*.

central electronic complex (CEC)
See *central processor complex (CPC)*.

central processor complex (CPC)
A physical collection of hardware that consists of channels, timers, main storage, and one or more central processors.

cluster
A loosely-coupled collection of independent systems, or *nodes*, organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster manager
The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager is the node with the lowest node number among the quorum nodes that are operating at a particular time.

compute node
A node with a mounted GPFS file system that is used specifically to run a customer job. ESS disks are not directly visible from and are not managed by this type of node.

CPC
See *central processor complex (CPC)*.

D

DA
See *declustered array (DA)*.

datagram
A basic transfer unit associated with a packet-switched network.

DCM
See *drawer control module (DCM)*.

declustered array (DA)
A disjoint subset of the pdisks in a recovery group.

dependent fileset
A fileset that shares the inode space of an existing independent fileset.

DFM
See *direct FSP management (DFM)*.

DHCP
See *Dynamic Host Configuration Protocol (DHCP)*.

direct FSP management (DFM)
The ability of the xCAT software to communicate directly with the Power Systems server's service processor without the use of the HMC for management.

drawer control module (DCM)
Essentially, a SAS expander on a storage enclosure drawer.

Dynamic Host Configuration Protocol (DHCP)
A standardized network protocol that is used on IP networks to dynamically distribute such network configuration parameters as IP addresses for interfaces and services.

E

Elastic Storage Server (ESS)
A high-performance, GPFS NSD solution made up of one or more building blocks that runs on IBM Power Systems servers. The ESS software runs on ESS nodes - management server nodes and I/O server nodes.

ESS Management Server (EMS)
An xCAT server is required to discover the I/O server nodes (working with the HMC), provision the operating system (OS) on the I/O server nodes, and deploy the ESS software on the management node and I/O server nodes. One management server is required for each ESS system composed of one or more building blocks.

encryption key
A mathematical value that allows components to verify that they are in communication with the expected server. Encryption keys are based on a public or private key pair that is created during the installation process. See also *file encryption key (FEK)*, *master encryption key (MEK)*.

ESS
See *Elastic Storage Server (ESS)*.

environmental service module (ESM)
Essentially, a SAS expander that attaches to the storage enclosure drives. In the case of multiple drawers in a storage enclosure, the ESM attaches to drawer control modules.

ESM
See *environmental service module (ESM)*.

Extreme Cluster/Cloud Administration Toolkit (xCAT)
Scalable, open-source cluster management software. The management infrastructure of ESS is deployed by xCAT.

F

failback
Cluster recovery from failover following repair. See also *failover*.

failover
(1) The assumption of file system duties by another node when a node fails. (2) The process of transferring all control of the ESS to a single cluster in the ESS when the other clusters in the ESS fails. See also *cluster*. (3) The routing of all transactions to a second controller when the first controller fails. See also *cluster*.

failure group
A collection of disks that share common access paths or adapter connection, and could all become unavailable through a single hardware failure.

FEK
See *file encryption key (FEK)*.

file encryption key (FEK)
A key used to encrypt sectors of an individual file. See also *encryption key*.

file system
The methods and data structures used to control how data is stored and retrieved.

file system descriptor
A data structure containing key information about a file system. This information includes the disks assigned to the file system (*stripe group*), the current state of the file system, and pointers to key files such as quota files and log files.

file system descriptor quorum
The number of disks needed in order to write the file system descriptor correctly.

file system manager
The provider of services for all the nodes using a single file system. A file system manager processes changes to the state or description of the file system, controls the regions of disks that are allocated to each node, and controls token management and quota management.

fileset
A hierarchical grouping of files managed as a unit for balancing workload across a cluster. See also *dependent fileset*, *independent fileset*.

fileset snapshot
A snapshot of an independent fileset plus all dependent filesets.

flexible service processor (FSP)
Firmware that provices diagnosis, initialization, configuration, runtime error detection, and correction. Connects to the HMC.

FQDN
See *fully-qualified domain name (FQDN)*.

FSP
See *flexible service processor (FSP)*.

fully-qualified domain name (FQDN)
The complete domain name for a specific computer, or host, on the Internet. The FQDN consists of two parts: the hostname and the domain name.

G

GPFS cluster
A cluster of nodes defined as being available for use by GPFS file systems.

GPFS portability layer
The interface module that each installation must build for its specific hardware platform and Linux distribution.

GPFS Storage Server (GSS)
A high-performance, GPFS NSD solution made up of one or more building blocks that runs on System x servers.

GSS
See *GPFS Storage Server (GSS)*.

H

Hardware Management Console (HMC)
Standard interface for configuring and operating partitioned (LPAR) and SMP systems.

HMC
See *Hardware Management Console (HMC)*.

I

IBM Security Key Lifecycle Manager (ISKLM)
For GPFS encryption, the ISKLM is used as an RKM server to store MEKs.

independent fileset
A fileset that has its own inode space.

indirect block
A block that contains pointers to other blocks.

inode
The internal structure that describes the individual files in the file system. There is one inode for each file.

inode space
A collection of inode number ranges reserved for an independent fileset, which enables more efficient per-fileset functions.

Internet Protocol (IP)
The primary communication protocol for relaying datagrams across network boundaries. Its routing function enables internetworking and essentially establishes the Internet.

I/O server node
An ESS node that is attached to the ESS storage enclosures. It is the NSD server for the GPFS cluster.

IP
See *Internet Protocol (IP)*.

IP over InfiniBand (IPoIB)
Provides an IP network emulation layer on top of InfiniBand RDMA networks, which allows existing applications to run over InfiniBand networks unmodified.

IPoIB
See *IP over InfiniBand (IPoIB)*.

ISKLM
See *IBM Security Key Lifecycle Manager (ISKLM)*.

J

JBOD array
The total collection of disks and enclosures over which a recovery group pair is defined.

K

kernel
The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

L

LACP
See *Link Aggregation Control Protocol (LACP)*.

Link Aggregation Control Protocol (LACP)
Provides a way to control the bundling of several physical ports together to form a single logical channel.

logical partition (LPAR)
A subset of a server's hardware resources virtualized as a separate computer, each with its own operating system. See also *node*.

LPAR
See *logical partition (LPAR)*.

M

management network
A network that is primarily responsible for booting and installing the designated server and compute nodes from the management server.

management server (MS)
An ESS node that hosts the ESS GUI and xCAT and is not connected to storage. It must be part of a GPFS cluster. From a system management perspective, it is the central coordinator of the cluster. It also serves as a client node in an ESS building block.

master encryption key (MEK)
A key that is used to encrypt other keys. See also *encryption key*.

maximum transmission unit (MTU)
The largest packet or frame, specified in octets (eight-bit bytes), that can be sent in a packet- or frame-based network, such as the Internet. The TCP uses the MTU to determine the maximum size of each packet in any transmission.

MEK
See *master encryption key (MEK)*.

metadata
A data structure that contains access information about file data. Such structures include inodes, indirect blocks, and directories. These data structures are not accessible to user applications.

MS
See *management server (MS)*.

MTU
See *maximum transmission unit (MTU)*.

N

Network File System (NFS)
A protocol (developed by Sun Microsystems, Incorporated) that allows any host in a network to gain access to another host or netgroup and their file directories.

Network Shared Disk (NSD)
A component for cluster-wide disk naming and access.

NSD volume ID
A unique 16-digit hexadecimal number that is used to identify and access all NSDs.

node
An individual operating-system image within a cluster. Depending on the way in which the computer system is partitioned, it can contain one or more nodes. In a Power Systems environment, synonymous with *logical partition*.

node descriptor
A definition that indicates how IBM Storage Scale uses a node. Possible functions include: manager node, client node, quorum node, and non-quorum node.

node number
A number that is generated and maintained by IBM Storage Scale as the cluster is created, and as nodes are added to or deleted from the cluster.

node quorum
The minimum number of nodes that must be running in order for the daemon to start.

node quorum with tiebreaker disks
A form of quorum that allows IBM Storage Scale to run with as little as one quorum node available, as long as there is access to a majority of the quorum disks.

non-quorum node
A node in a cluster that is not counted for the purposes of quorum determination.

O

OFED
See *OpenFabrics Enterprise Distribution (OFED)*.

OpenFabrics Enterprise Distribution (OFED)
An open-source software stack includes software drivers, core kernel code, middleware, and user-level interfaces.

P

pdisk
A physical disk.

PortFast
A Cisco network function that can be configured to resolve any problems that could be caused by the amount of time STP takes to transition ports to the Forwarding state.

R

RAID
See *redundant array of independent disks (RAID)*.

RDMA
See *remote direct memory access (RDMA)*.

redundant array of independent disks (RAID)
A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

recovery
The process of restoring access to file system data when a failure has occurred. Recovery can involve reconstructing data or providing alternative routing through a different server.

recovery group (RG)
A collection of disks that is set up by IBM Storage Scale RAID, in which each disk is connected physically to two servers: a primary server and a backup server.

remote direct memory access (RDMA)
A direct memory access from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively-parallel computer clusters.

RGD
See *recovery group data (RGD)*.

remote key management server (RKM server)
A server that is used to store master encryption keys.

RG
See *recovery group (RG)*.

recovery group data (RGD)
Data that is associated with a recovery group.

RKM server
See *remote key management server (RKM server)*.

S

SAS
See *Serial Attached SCSI (SAS)*.

secure shell (SSH)
A cryptographic (encrypted) network protocol for initiating text-based shell sessions securely on remote computers.

Serial Attached SCSI (SAS)
A point-to-point serial protocol that moves data to and from such computer storage devices as hard drives and tape drives.

service network
A private network that is dedicated to managing POWER8® servers. Provides Ethernet-based connectivity among the FSP, CPC, HMC, and management server.

SMP
See *symmetric multiprocessing (SMP)*.

Spanning Tree Protocol (STP)
A network protocol that ensures a loop-free topology for any bridged Ethernet local-area network. The basic function of STP is to prevent bridge loops and the broadcast radiation that results from them.

SSH
See *secure shell (SSH)*.

STP
See *Spanning Tree Protocol (STP)*.

symmetric multiprocessing (SMP)
A computer architecture that provides fast performance by making multiple processors available to complete individual processes simultaneously.

T

TCP
See *Transmission Control Protocol (TCP)*.

Transmission Control Protocol (TCP)
A core protocol of the Internet Protocol Suite that provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating over an IP network.

V

VCD
See *vdisk configuration data (VCD)*.

vdisk
A virtual disk.

vdisk configuration data (VCD)
Configuration data that is associated with a virtual disk.

X

xCAT
See *Extreme Cluster/Cloud Administration Toolkit*.

Index

Special Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

Product Number: 5765-DME
5765-DAE
5737-F34

SC28-3452-01